Chapter 6
Proximal Methods for Image Processing

An Introduction to ProxToolbox for Testing Algorithms on the Göttingen Datasets

D. Russell Luke

Was ist dann eigentlich ein Hilbertischer Raum?
– David Hilbert to John von Neumann [1]

6.1 All Together Now

A major challenge in building and maintaining collaborations across disciplines is to establish a common language. Sounds simple enough, but even a common language is not helpful without a common understanding of the basic elements. When it comes to the day-to-day exchange of data and software, this means building a common data management and processing environment. Try to do this, however, and you learn very quickly that even for something as concrete as building software that everyone can use, there are different ways of interpreting and understanding what the software does. In the context of X-ray diffraction, for instance, what a physicist might understand as a software routine that simulates the propagation of an X-ray through an optical device, a mathematician would understand as an operator with certain mathematical properties.

The first successful algorithms for phase retrieval were developed and understood by physicists as iterative procedures that simulate the forward and backward propagation of a wave through an optical device, where in each iteration the computed wave is adjusted to fit either measurement data or some experimental constraint, like the shape of the aperture or the illuminating beam. Later, mathematicians reinterpreted these operations in terms of the application of *projectors* to iterates of a *fixed point mapping*. Of most recent vintage is an effort by a new generation of applied mathematicians to sidestep the more interesting aspects of the physicists'

D. R. Luke (✉)
Institute for Numerical and Applied Mathematics, Universität Göttingen,
Lotzestrasse 16-18, 37083 Göttingen, Germany
e-mail: r.luke@math.uni-goettingen.de

T. Salditt et al. (eds.), *Nanoscale Photonic Imaging*, Topics in Applied Physics 134,
https://doi.org/10.1007/978-3-030-34413-9_6

algorithms—namely that they occasionally don't work—by lifting the problem to a space that is too high-dimension for any practical purposes, and then *relaxing* the underlying problem to something with theoretically nicer properties, but whose solution bears little meaningful relationship to the problem at hand. At this point, one is reminded of Richard Courant's lamentation, "the broad stream of scientific development may split into smaller and smaller rivulets and dry out." [2] Here's to swimming against the current.

6.1.1 What Seems to Be the Problem Here?

The story of computational phase retrieval in X-ray imaging is a perfect example of how different communities can come to understand the same things in very different ways. This serves as a sort of origin story for the ProxToolbox [3]

```
http://num.math.uni-goettingen.de/proxtoolbox/
```

which is the subject of this chapter. The setting for phase retrieval has been presented in Chap. 2 and this will be the stuff cooked in the mathematical crucible that the ProxToolbox represents.

The measurements are intensity readings which are denoted simply as a nonnegative-valued vector I, with n elements corresponding to the pixels in the CCD array (see 2.12). The model for these measurements is developed in Sect. 2.1. The various modalities (near field/far field) have the general form

$$\left\|\left(\mathcal{D}_F(\psi)\right)(x_k, y_j)\right\| = \sqrt{I(x_k, y_j)}, \quad (k = 1, 2, \ldots, n_x)\,(j = 1, 2, \ldots, n_y). \tag{6.1}$$

In (2.12) the model is given in the continuum with the intensity $I(x, y)$ at the position (x, y) in the measurement plane. The actual measurements consist of pixels indexed by $k = 1, 2, \ldots, n_x$ and $j = 1, 2, \ldots, n_y$ corresponding to positions (x_k, y_j) in the measurement plane. The model then represents a system of $n \equiv n_x n_y$ equations in $2n$ unknowns, the real and imaginary part of $\mathcal{D}_F(\psi)$ at each of the n pixels. With this discretization, the mapping $\mathcal{D}_F$ is understood as a discrete *Fresnel propagator* with Fresnel number F (see 2.84). To make things simpler, the indexes k and j are combined uniquely into a single index $i = 1, 2, \ldots, n$ so that the data model is just

$$\left\|\left(\mathcal{D}_F(\psi)\right)_i\right\| = \sqrt{I_i}, \quad \forall\ i = 1, 2, \ldots, n. \tag{6.2}$$

The solution to the phase retrieval problem as presented here is the complex-valued vector $\psi \in \mathbb{C}^n$ that satisfies (6.2) or, more generally (6.3). With only a single measurement the problem is *underdetermined*, that is, too many unknowns and not enough equations. To further constrain the problem, there are a number of possibilities. First, one could (and should) include a priori information implicit in the experiment (support, nonnegativity, sparsity, etc). The next thing one could try is to adjust the instrument in some known fashion and take several measurements of the same object. The mapping $\mathcal{D}_F$ is the model for the mapping of the electromagnetic field at the object to the field where the intensity is measured, either near field or far field. For the purpose of this tutorial, these are mappings from vectors in $\mathbb{C}^n$ to vectors in $\mathbb{C}^n$. This can be modified in a number of different ways. Representing n-dimensional complex-valued vectors instead as two-dimensional vectors on an n-dimensional product space, the phase retrieval problem is to find $\psi = (\psi_1, \psi_2, \dots, \psi_n) \in \left(\mathbb{R}^2\right)^n$ ($\psi_i \in \mathbb{R}^2$) satisfying (6.2) for all i in addition to qualitative constraints and/or additional measurements.[1]

Ptychography was briefly mentioned in Chap. 2. Ptychography is harder to say than to describe. A quilt is a ptychogram of sorts. Or put in more technical terms, ptychography is a combination of blind deconvolution and computed tomography for phase retrieval. The original idea proposed by Hegerl and Hoppe [4], was just the computed tomography part: to stitch together the original object ψ from many measurements at different settings of the instrument, modeled by $\mathcal{D}_j$, the j indexing the setting. One of the implicit complications of conventional ptychography, which differs from the original is that the illuminating beam is also unknown—this is the blind deconvolution part.

In the above, different Fresnel numbers correspond to collecting the intensities I_i at different planes orthogonal to the direction of propagation. This further constrains the problem. The model is only a little more involved than simple phase retrieval (6.2). For m different intensity measurements, each consisting of n pixels, the generalized ptychography/phase diversity model takes the form

$$\| \left(\mathcal{D}_j(\psi)\right)_i \| = \sqrt{I_{i,j}} \quad \forall i = 1, 2, \dots, n, \forall j = 1, 2, \dots m. \tag{6.3}$$

This fits the first ptychographic reconstruction procedure [4] which assumed that the illuminating beam, characterized by $\mathcal{D}_j$, was known. In the *blind* ptychography problem – analogous to the blind deconvolution—the beam is not completely known. This corresponds to what is commonly understood by ptychography in modern applications [5–8]. To account for the unknown beam characteristics, the mapping $\mathcal{D}_j$ is further decomposed:

$$\mathcal{D}_j(z, \psi) \equiv \mathcal{F}\left(S_j\,(z) \odot \psi\right) \tag{6.4}$$

Here $\mathcal{F}$ is a parameter-free propagator and $S_j : \mathbb{C}^n \to \mathbb{C}^n$ denotes the j-th linear operator representing some known adjustment to the beam—a lateral shift, or translation in the direction of propagation—$z \in \mathbb{C}^n$ is the unknown vector characterizing the probe, and $\odot$ is the elementwise Hadamard product.

The problem in blind ptychography is to reconstruct simultaneously the object ψ and illuminating beam u from a given ptychgraphic dataset. Near-field, or *in-line*

[1]The issue of whether to represent the vectors as points in $\mathbb{R}^{2n}$ or points in $\mathbb{C}^n$ is notational. The representation as points in $\mathbb{C}^n$ is more convenient for the purpose of explanation, but on a computer you will need to work with $\mathbb{R}^{2n}$.

ptychography [9] involves moving the imaging plane along the axis of propagation of the beam (i.e., away from the plane where the object lies). This is very similar to *phase diversity* in astronomy [10], but there one changes the *defocus* in the far-field instead of the imaging plane in the near field (also, one does not have to recover the beam in astronomy applications). Mathematically, however, the two instances have the same structure. In conventional far-field ptychographic experiments the beam is much smaller than the specimen. The different measurements consist of scans of ψ in the lateral direction with sufficient overlap between successive images. Lateral translations and translations along the axis of propagation were combined in [11]. Note that the last case is least restrictive in terms of probe properties, see also [12] for a detailed comparison and discussion.

The issue of *existence* of a ψ that satisfies all the equations is discussed in Chap. 23. For this chapter, existence is recast as *consistency* of the measurements and the physical model. The data is exact.[2] What is not entirely accurate is the model for the data and the *computational bandwidth*, e.g. finite precision arithmetic.

Though it might seem unintuitive, for algorithmic reasons, it is better to separate the aspects of the imaging model having to do with the field at the object plane from those having to do with the field at the image plane. Denote $\mathbf{u} \equiv (u_1, u_2, \dots, u_m)$ with $u_j \in \mathbb{C}^n$ $(j = 1, 2, \dots, m)$ and define the measurement sets

$$M_j \equiv \left\{u \in \mathbb{C}^n \mid \left\|(\mathcal{F}u)_i\right\| = \sqrt{I}_{j,i},\ (i = 1, 2, \dots, n)\right\} \quad (j = 1, 2, \dots, m) \tag{6.5}$$

where $\mathcal{F}$ is a parameter-free propagator in (6.4). The sets M_j are nothing more than the *phase sets*, or the set of all vectors that *could* explain the data. Solutions to the most general physical model represented by (6.3) consists of any triple of vectors $(z, \psi, \mathbf{u}) \in \mathbb{C}^n \times \mathbb{C}^n \times (\mathbb{C}^n)^m$ in the set

$$\mathcal{M} \equiv \left\{(z, \psi, \mathbf{u}) \in \mathbb{C}^n \times \mathbb{C}^n \times \mathbb{C}^{n\times m} \mid S_j(z) \odot \psi = u_j,\ \ j = 1, 2, \dots, m\right\} \tag{6.6}$$

such that $u_j \in M_j$ and the beam z and object ψ satisfy any other reasonable qualitative constraints. The constraints on the unknowns z, ψ and $\mathbf{u}$ are separable and given by

$$X \equiv \{\text{qualitative constraints on the probe}\}, \tag{6.7a}$$
$$O \equiv \{\text{qualitative constraints on the specimen}\}, \tag{6.7b}$$
$$M \equiv M_1 \times M_2 \times \cdots \times M_m, \text{ (measurement constraints)}. \tag{6.7c}$$

As before, the qualitative constraints characterized by X and O are support, support-nonnegativity or magnitude constraints corresponding respectively to whether the illumination and specimen are supported on a bounded set, whether these (most likely only the specimen) are "real objects" that somehow absorb or attenuate the

[2]This is a minority opinion, but it seems to be the height of hubris to think that an empirical observation is an approximation to a theoretical model rather than the other way around. The only thing that is indisputable is that the instrument behavior and the predicted behavior don't match up as well as desired.

probe energy, or whether these are "phase objects" with a prescribed intensity but varying phase. A support constraint for the set X, for instance, would be represented by

$$X \equiv \left\{ z = (z_1, z_2, \dots, z_n) \in \mathbb{C}^n \mid |z_i| \le R \text{ and for } i \notin \mathbb{I}_X \; z_i = 0 \right\}, \tag{6.8}$$

where $\mathbb{I}_X$ is the index set corresponding to which pixels in the field of view the probe beam illuminates and R is some given amplitude.

6.1.2 What Is an Algorithm?

One of the best known algorithms for phase retrieval is Fienup's Hybrid Input-Output algorithm (HIO) [13] discussed in 2.88. This algorithm illustrates just about every difficulty one encounters in collaborating across disciplines, and sets the stage for the rest of the tutorial below. In the present setting, HIO with a support constraint is given as

$$(\forall k \in \mathbb{N})(\forall i = 1, 2, \dots, n) \;\; \psi_i^{k+1} = \begin{cases} \left(\mathcal{P}_M(\psi^k)\right)_i, & \text{if } i \in D; \\ \psi_i^k - \beta_k\left(\mathcal{P}_M(\psi^k)\right)_i, & \text{otherwise.} \end{cases} \tag{6.9}$$

Here $\mathbb{N}$ is the set of counting numbers, D indicates an index set where it is imagined that some constraints are satisfied; the mapping $\mathcal{P}_M$ fits the current iterate ψ^k to the data by propagating this guess through a model optical system, fixing the amplitude at the measurement surface to match the observed intensity and then propagating the resulting field back to the plane of the object. Putting (2.84) in into the present context, with the set M consisting of just a single intensity image, $M \equiv M_1$, yields

$$\psi_+ \simeq \mathcal{P}_M(\psi) \simeq \mathcal{D}_{-F}(\widehat{\psi}), \quad \widehat{\psi}_i \simeq \sqrt{I_i}\,\frac{(\mathcal{D}_F(\psi))_i}{\left\|(\mathcal{D}_F(\psi))_i\right\|} \tag{6.10}$$

The symbol $\simeq$ indicates that this is not really and equivalence relation: for the moment think of it as equivalence with exceptions.

One obvious exception is when $(\mathcal{D}_F(\psi))_i = 0$. If you are a physicist you might argue that $(\mathcal{D}_F(\psi))_i = 0$ on a set of measure zero—a fancy way for saying never, with infinite precision arithmetic. Except that electronic processors operate with finite precision arithmetic and zero is therefore enormous on a computer. In fact, with double precision, zero is not smaller than $1e-16$. To see what kind of error this can lead to, suppose that $(\mathcal{D}_F(\psi))_i = -1e-15$ with infinite precision arithmetic, but because of roundoff, the computer returns $1e-15$. A very small difference locally. But suppose that, at this pixel, the measured intensity is $I_i = 10$. In computing the projection, the computer returns a point with $\widehat{y}_i = 10$ instead of $\widehat{y}_i = -10$. This makes an enormous difference. The typical user won't see this kind of error often, but in numerical studies in [14] it happened about 12% of the time, which is not insignificant. Anyone with experience in programming knows to be careful about dividing. Without thinking much more about it, one usually codes some exception to avoid problems when division gets too dicey.

But there is another reason to pay attention to this exceptional point: *convexity*. The mathematical understanding of this and other phase retrieval algorithms originating from the optics community begins with viewing the entire collection of points satisfying the data and the qualitative constraints as *sets* and then viewing the operations in the above iterative procedures as *metric projections* onto these sets. A metric projector of a point ψ onto a set C is simply an operator that maps ψ to all points in C that are nearest to ψ. The operation $\mathcal{P}_M$ in (6.10) was long called a projection in the optics literature, but it was not shown to be a *metric projector* until [14, Corollary 4.3] where it was pointed out that the projector is *set-valued* in general. Being more careful one *should* write

$$\psi_+ \in \mathcal{P}_M(\psi) = \left\{ \mathcal{D}_{-F}(\widehat{\psi}) \;\middle|\; \widehat{\psi}_i \in \begin{cases} \sqrt{I_i}\frac{(\mathcal{D}_F(\psi))_i}{\|(\mathcal{D}_F(\psi))_i\|}, & \text{if } (\mathcal{D}_F(\psi))_i \neq 0, \\ \sqrt{I_i}\mathbb{S}, & \text{if } (\mathcal{D}_F(\psi))_i = 0. \end{cases} \right\} \tag{6.11}$$

The symbol $\mathbb{S}$ denotes the unit sphere in the complex plane ($\mathbb{R}^2$) and $\sqrt{I_i}\mathbb{S}$ is the sphere of radius $\sqrt{I_i}$. The symbol $\in$ is a reminder that the right hand side is a set of elements, and the left hand side is just a selection—any selection—from this set. The change in notation from "=" to "$\in$" is not just pedantic nit-picking, but underscores the fact that the problem is fundamentally *nonconvex*, meaning that you can find two points in the set where the line segment joining the two points leaves the set. In (6.11) take two points on opposite ends of the sphere $\sqrt{I_i}\mathbb{S}$: the line joining them is not in $\sqrt{I_i}\mathbb{S}$. If the sets M were convex (line segments joining any two points in the set are contained in said set), then the projector would be single-valued and one could forget the whole technicality, not to mention any worries about numerical instability.

Returning to iterative algorithms just in the context of phase retrieval (the probe z is known and there is only a single measurement $M = M_1$ so that the variables $\mathbf{u}$ are not needed), the procedure (6.9) is a natural way to think of a numerical procedure when approaching things physically: one makes a guess for the object ψ^0, propagates it through the optical system according the model given by (6.10) and updates this guess to ψ^1 depending on whether the elements satisfy the data and some a priori constraint. And repeat. The user would stop the iteration either when he needed to go for coffee or when the iterates stop making progress, in some loosely defined way. The subtle point here is that (6.9) is not a *fixed point iteration*, but rather a simulation of a physical process.

Bauschke, Combettes and Luke [15] showed that the HIO algorithm (6.9) is equivalent to

$$\psi^{k+1} \in \tfrac{1}{2}\big(\mathcal{R}_{\mathfrak{S}}(\mathcal{R}_M + (\beta_k - 1)\mathcal{P}_M) + \mathrm{Id} + (1-\beta_k)\mathcal{P}_M\big)(\psi^k). \tag{6.12}$$

where Id is the identity mapping (does nothing to the point), $\mathfrak{S}$ is the set of vectors satisfying a bounded support constraint, and

$$\mathcal{R}_M \equiv 2\mathcal{P}_M - \mathrm{Id} \quad \text{and} \quad \mathcal{R}_{\mathfrak{S}} \equiv 2\mathcal{P}_{\mathfrak{S}} - \mathrm{Id} \tag{6.13}$$

with

$$(\mathcal{P}_{\mathfrak{S}}\psi)_j = \begin{cases} \psi_j & \text{if } j \in D \\ 0 & \text{else.} \end{cases} \tag{6.14}$$

The mappings $\mathcal{R}_{\mathfrak{S}}$ and $\mathcal{R}_M$ are called *reflectors* because they send points to their reflection points on the opposite side of the set onto which one projects. The iteration (6.12) is the Hybrid Projection Reflection (HPR) algorithm proposed in [15]. When $\beta_n = 1$ for all n, then the iteration takes the form [16]

$$\psi^{k+1} \in \tfrac{1}{2}\big(\mathcal{R}_{\mathfrak{S}}(\mathcal{R}_M + \mathrm{Id}\big)(\psi^k). \tag{6.15}$$

This is the popular *Douglas-Rachford Algorithm* [17] following the formulation of Lions and Mercier in the context of *monotone operator equations* [18]. In both cases, the desired point is a *fixed point* of the mapping T, that is a point $\overline{\psi}$ such that $\overline{\psi} = T\overline{\psi}$ where either $T = \frac{1}{2}\big(\mathcal{R}_{\mathfrak{S}}(\mathcal{R}_M + (\beta_k - 1)\mathcal{P}_M) + \mathrm{Id} + (1-\beta_k)\mathcal{P}_M\big)$ or $T = \frac{1}{2}\big(\mathcal{R}_{\mathfrak{S}}(\mathcal{R}_M + \mathrm{Id}\big)$.

Whether or not the iterations above converge to a fixed point, and what this point has to do with the problem at hand is the subject of Chap. 23. For the purposes of this tutorial, the algorithm will simply be run with the given mappings and the user will be left to interpret the result. A few tools are provided within the ProxToolbox to monitor the iterates according to mathematical, as opposed to physical, criteria. For the beginning reader all that is important to keep in mind is that, first of all, the algorithms don't always converge, and second of all, when they converge, the limit point is *not* a solution to the problem you thought you were solving, but you can usually get there easily from the limiting fixed point. Another issue to keep in mind is that the physical criteria that scientists apply to judge the quality of a computed solution usually does not correspond to the mathematical criteria used to characterize and quantify convergence of an algorithm. It is not uncommon to see pictures of "solutions" returned by various algorithms at iteration k, or to see a comparison of a root mean-squared error estimate of an iterate of various algorithms. This is, from a mathematical perspective, not really meaningful for several reasons. The first reason is that, unless the underlying optimization problem is to minimize the root mean-squared error of something, there is no reason to expect that an algorithm should do this. The second reason is that, as already mentioned, the iterates of some algorithms, like Douglas-Rachford, are not the points that approximate solutions to the desired optimization problem, but their *shadows*, defined as the projection of these points onto a relevant set, are. Comparison of the quality of solutions returned by algorithms is common, but it should be recognized that such comparisons are not mathematical, but rather *phenomenological*, if of any scientific significance at all.

The identification of the HIO algorithms with Douglas-Rachford, at least for one parameter value, made a lot of sense when it was first discovered. The HIO algorithm is famously unstable. The way most people use it today is to get themselves in the neighborhood of a good solution by running 10–40 iterations of HIO, at which point they switch to a more stable algorithm to clean up their images. The value of HIO or Douglas-Rachford is that they rarely get stuck in local minimums. The identification with Douglas-Rachford makes this phenomenon clear since it can be proved that, if the sets $\mathfrak{S}$ and M do not intersect, then Douglas-Rachford *does not possess fixed points*. If M were convex, then you could even prove that the iterates must *diverge to infinity* in the direction of the gap vector between best approximation pairs between the sets [19]. For nonconvex problems like noncrystallographic phase retrieval, the iterates need not diverge, but they *cannot* converge. In Chap. 23 the convergence theory is discussed in some detail.

As should be clear by now, there is no equation being solved here, but rather some point is sought, any point, that satisfies an equation and any other kind of requirement one might like to add. It is high time to bring the main character of this story to the stage. In the most general format (ptychography) this has the form

General Problem

$$\text{Find } (z, \psi, \mathbf{u}) \in \mathcal{M} \cap (X \times O \times M)\,.$$

This is a *feasibility problem* and what algorithms like (6.12) and (6.15) aim to solve, if possible. For the moment, it is easiest to examine the more elementary phase retrieval problem (ptychography with one measurement):

$$\text{Find } \psi \in \mathfrak{S} \cap M. \tag{6.16}$$

The sets $\mathfrak{S}$ and M have the explicit characterizations

$$M \equiv \left\{ \psi \in \mathbb{C}^n \;\middle|\; \left|(\mathcal{D}_F \psi)_i\right| = \sqrt{I_i},\ i = 1, 2, \ldots, n \right\} \tag{6.17}$$

and

$$\mathfrak{S} \equiv \left\{ \psi \in \mathbb{C}^n \mid \psi_i = 0 \text{ if } i \notin D \right\}. \tag{6.18}$$

The projectors onto these sets are given by (6.11) and (6.14). When the requirements become so narrow that no point can satisfy all of them, the problem is said to be *inconsistent*. The reason for the instability of HIO and Douglas-Rachford lies with the failure of the sets $\mathfrak{S}$ and M to have points in common. This indicates a fundamental *inconsistency* of the physical model.

The behavior of these algorithms in the presence of model inconsistency tends to be mistaken for another bugbear of inverse problems, namely *nonuniqueness*. For example, around the turn of the millennium, oversampling in the image domain was proposed to overcome the nonuniqueness of phase reconstructions for noncrystallographic observations [20]. A few moments reflection on elementary Fourier analysis, and careful reading of Chap. 2 is all you need to convince yourself, however, that oversampling has less to do with uniqueness than with inconsistency. Increased, but still finite, sampling in the image domain just pushes the inconsistency, or gap, between the sets $\mathfrak{S}$ and M to some level below either your numerical or experimental precision. It might look like the iteration has converged, but what has really happened is that the movement of the iterates has become so small that it is no longer detectable with a fixed arithmetic precision. This is just a physical manifestation of the fact from Fourier analysis that objects with compact support do not have compactly supported Fourier transforms, and vice versa. Since the measurements are finite, the object that is recovered cannot be finite. This means that the only time phase retrieval can be *consistent* is when imaging periodic crystals.

To be sure, uniqueness is nice when you have it, but you first need to clear up the issue of uniqueness *of what. No wave ψ with compact support can generate the given intensity I.* When the problem is inconsistent, it suffices to find some point, any point, that comes as close to both sets as possible. This is a *best approximation problem* and takes the form of the usual optimization problem:

$$\begin{array}{ll} \underset{\psi\in\mathbb{C}^n}{\text{minimize}} & \frac{\lambda}{2(1-\lambda)}\,\mathrm{dist}^2(\psi,\mathfrak{S}) \\ \text{subject to} & \psi\in M \end{array}. \tag{6.19}$$

The reason for minimizing the distance squared instead of the distance is to have a nice smooth objective function—it doesn't really change the problem. Nor does the factor $\frac{\lambda}{2(1-\lambda)}$ out front, but it has a huge impact on the next algorithm, which solves (6.19). So the question of uniqueness amounts to whether problem (6.19) has a unique solution. Experts in optimization don't often worry about uniqueness, but rather the existence of *local minima* to (6.19). This is one of the few remaining unresolved mathematical issues in phase retrieval.

While, in most applications, the projections onto the sets X, O and M in (6.7) have a closed form and can be computed very accurately and efficiently, there does not seem to be any method, analytic or otherwise, for computing the projection onto the set $\mathcal{M}$ defined by (6.6). This might be another good reason for avoiding a feasibility model. Indeed, if the projections are too difficult, or impossible to compute analytically, then the large part of the advantage of projection methods evaporates. Nevertheless, this essentially two-set feasibility model suggests a wide range of techniques within the family of projection methods, alternating projections, averaged projections and Douglas–Rachford being representative members. In contrast to these, methods based on optimization models can avoid the difficulty of computing a projection onto the set $\mathcal{M}$ by instead minimizing a nonnegative coupling function that takes the value 0 (only) on $\mathcal{M}$. The model for this is a *constrained least squares minimization* model

$$\text{Find } \left(\overline{z}, \overline{\psi}, \overline{\mathbf{u}}\right) \in \operatorname{argmin} \left\{\mathbf{F}(z, \psi, \mathbf{u}) \mid (z, \psi, \mathbf{u}) \in X \times Y \times M\right\} \tag{6.20}$$

where

$$\mathbf{F}(z, \psi, \mathbf{u}) \equiv \sum_{j=1}^{m} \|S_j(z) \odot \psi - u_j\|^2. \tag{6.21}$$

What has happened here is that the set $\mathcal{M}$ defined by (6.6) has been replaced by the least squares objective to avoid the complication of computing the projection onto $\mathcal{M}$.

The relaxed averaged alternating reflections algorithm (RAAR[3]) first proposed in [21] addresses the instability of the Douglas-Rachford algorithm by anchoring the usual Douglas-Rachford iterates to one of the sets:

$$\psi^{k+1} \in \left(\tfrac{\lambda}{2}\left(\mathcal{R}_{\mathfrak{S}}(\mathcal{R}_M + \mathrm{Id}\right) + (1-\lambda)\mathcal{P}_M\right)(\psi^k). \tag{6.22}$$

When $\lambda \in [0, 1)$ it was shown in [22] that this algorithm is equivalent to the Douglas-Rachford algorithm applied to (6.19). For the moment, just recognise that this is a convex combination of (6.15) with the projection onto the set M. If one wanted to play around further, the constraints $\mathfrak{S}$ and M can be changed without changing the form of the fixed point iterations. For instance, if the thing one is trying to recover, ψ, is actually an *electron density*, it should be a real-valued, positive vector; so instead of the set $\mathfrak{S}$, one should restrict the possible points to

$$\mathfrak{S}_+ \equiv \left\{\psi \in \mathbb{C}^n \;\middle|\; \psi_i = \begin{cases}\max\{0, \Re(\psi_i)\} & \text{if } i \in D\\ 0 & \text{else}\end{cases}\right\}. \tag{6.23}$$

The RAAR algorithm then takes the form

$$\psi^{k+1} \in \left(\tfrac{\lambda}{2}\left(\mathcal{R}_{\mathfrak{S}_+}(\mathcal{R}_M + \mathrm{Id}\right) + (1-\lambda)\mathcal{P}_M\right)(\psi^k),$$

which is hardly a change from (6.22). In fact, mathematically there is no qualitative difference between the two. When translated back to the format of the original HIO algorithm, this takes the form [21, Prop.2.1][4]: $(\forall k \in \mathbb{N})($ for $i = 1, 2, \ldots, n)$,

[3]The names for these algorithms have evolved since their first introduction. In [19] the procedure that is today known as the Douglas-Rachford algorithm was called the Averaged Alternating Reflections algorithm, which then explains the genesis of the name RAAR for (6.22). Since Douglas-Rachford is more or less the accepted name for (6.15), (6.22) is called DRλ in more recent matheamtical articles. Nevertheless, RAAR is more common in the physics literature, so that is the nomenclature used here. In the ProxToolbox, however, the DRλ nomenclature is used.

[4](6.24) corrects a sign error in the lower half of [21, Eq (14)]

$$\psi_i^{k+1} = \begin{cases} \left(\mathcal{P}_M(\psi^k)\right)_i, & \text{if } i \in D \text{and} \left(\mathcal{R}_M(\psi^k)\right)_i \geq 0; \\ \lambda\psi_i^k + (1-2\lambda)\left(\mathcal{P}_M(\psi^k)\right)_i, & \text{otherwise.} \end{cases} \tag{6.24}$$

Comparing (6.24) with (6.9), it is clear that these are very different algorithms. If the object domain constraint were to change, (for instance, to a magnitude constraint in the object plane for a complex-valued object) the physical description analogous to (6.24) would change dramatically yet again, but the description as a fixed point mapping would always have the form

$$\psi^{k+1} \in T_{RAAR}\psi^k \equiv \left(\tfrac{\lambda}{2}\left(\mathcal{R}_O(\mathcal{R}_M + \mathrm{Id}\right) + (1-\lambda)\mathcal{P}_M\right)(\psi^k) \tag{6.25}$$

where O is a placeholder for the constraint in the physical domain (see also (2.89)). The main point here is that the mathematical properties of the fixed point mapping T_{RAAR} depend on the properties of the sets M and O, but the algorithm is always the same.

From this point hence, the word *algorithm* will be used more or less synonymously with the phrase *fixed point iteration*. This will be a convenient way to pack several (hundreds of) lines of code into a single symbol T, for the *fixed point mapping*. This T takes a guess x^0 and replaces it with an update x^1. In mathematical terms, T *maps* $x^0 \in X$ to $x^1 \in X$ where X is the *domain* and *image* spaces of T, the shorthand for which is $T : X \to X$. The domain and image spaces need not be the same, but for fixed point iterations they are. One important feature of this way of thinking about things is that the guess and the update are the same kinds of objects with the same physical interpretation. This is different than a *function* or more generally a *relation* which can map a point to anything, for instance a number, a color, or a set. A *fixed point iteration* is the process of repeatedly applying the fixed point mapping T: given x^0, generate a sequence of points x^k via

$$(\forall k \in \mathbb{N}) \qquad x^{k+1} = Tx^k. \tag{6.26}$$

There are a number of accessories one can add to (6.26). These take the form: given x^0, and an update rule $\mathcal{A}_k$ ($k = 1, 2, \dots$), generate a sequence of points x^k via

$$(\forall k \in \mathbb{N}) \qquad \begin{aligned} y^{k+1} &= Tx^k; \\ x^{k+1} &= \mathcal{A}_k\left(x^k, y^{k+1}\right). \end{aligned} \tag{6.27}$$

The main difference between (6.26) and (6.27) is that in the latter the operations from one iteration to the next can evolve and adjust along with the iterations. These are invariably called *accelerations* because that is the name of the game.

6.1.3 What Is a Proximal Method?

A *proximal method* is a fixed point iteration (6.26) or its acceleration (6.27) where the fixed point mapping T consists of *proximal mappings*. A proximal mapping has a specific mathematical definition, but before giving this an intuitive description will probably be more helpful. A proximal mapping sends a point to one or more points that strike a balance between solving a minimization problem and staying close to the original point. The projectors of the previous section are proximal mappings. To see this, consider the function

$$\iota_{\Omega}(u) \equiv \begin{cases} 0, & \text{if } u \in \Omega, \\ +\infty, & \text{else.} \end{cases} \tag{6.28}$$

where Ω is some set. Allowing this function to take the value $+\infty$ is very convenient. The optimization problem corresponding to minimizing ι_Ω while staying as close as possible to $\overline{u}$ is

$$\underset{u}{\text{minimize}}\ \iota_\Omega(u) + \tfrac{1}{2\lambda}\|u - \overline{u}\|^2,$$

and the *solution* to this problem is written

$$\operatorname{argmin}_u \left\{\iota_\Omega(u) + \tfrac{1}{2\lambda}\|u - \overline{u}\|^2\right\}.$$

The parameter $\lambda > 0$ will become important in a moment, but it has no significance in this context since the solution to the optimization problem above is the same for all positive values of λ. The solution to this problem is the *set* of points in Ω that are nearest to $\overline{u}$, or the set $\mathcal{P}_\Omega\overline{u}$. This should not be confused with the *optimal value* of this problem, which in this context is just the distance of the point $\overline{u}$ to the set Ω. For practical purposes one simply takes a selection from the set; this is denoted $u^+ \in \mathcal{P}_\Omega\overline{u}$ and u^+ is called a *projection*.

The function ι_Ω is not the only function one could use, hence the more general use of the terminology *proximal mapping* for the general function f [23]

$$\operatorname{prox}_{f,\lambda}(\overline{u}) \equiv \operatorname{argmin}_u \left\{f(u) + \tfrac{1}{2\lambda}\|u - \overline{u}\|^2\right\}. \tag{6.29}$$

Here the value of λ plays the role of dialing up or down the requirement of staying close to the point $\overline{u}$. This is often understood as a *step-length parameter* in the context of algorithms: the smaller λ is, the greater the penalty for moving away from $\overline{u}$.

The algorithm (6.15) written using the formalism of proximal mappings takes the form

$$\psi^{k+1} = \tfrac{1}{2}\left(\mathcal{R}_{f_0,\lambda_0}\mathcal{R}_{f_1,\lambda_1} + \mathrm{Id}\right)(\psi^k). \tag{6.30}$$

Here $\mathcal{R}_{f_0,\lambda_0}$ and $\mathcal{R}_{f_1,\lambda_1}$ are called *proximal reflectors* defined by

$$\mathcal{R}_{f_0,\lambda_0} \equiv 2\operatorname{prox}_{f_0,\lambda_0} - \mathrm{Id} \quad \text{and} \quad \mathcal{R}_{f_1,\lambda_1} \equiv 2\operatorname{prox}_{f_1,\lambda_1} - \mathrm{Id}. \tag{6.31}$$

This is quite liberating because now, the same basic fixed point mapping can be applied without any changes in the mathematical theory to a much broader range of problem types.

It is worthwhile spending a few moments to marvel at prox. This is a mapping from points in a space X to points in the same space; using mathematical notation $\text{prox}_{f,\lambda} : X \to X$. But, look again at (6.29): this is the solution to another optimization problem. There are two important things to notice about this observation, first of which is that a *mapping* has been created out of an optimization problem. This is what a mathematician might call pretty. The second thing to notice is that the *value* of the optimization problem—"the answer to the ultimate question of life, the universe and everything" [24]—is beside the point.[5]

6.1.4 On Your Mark. Get Set...

There are three groups of readers envisioned for this tutorial. The first group is students, of either physics or mathematics, wishing to get hands-on numerical experience with classical algorithms for real-world problems in the physical sciences. The second group is optical scientists who already know what they want to do, but would like a repository of algorithms to see what works for their problem and what does not work. The third group is applied mathematicians who have new algorithmic ideas, but need to see how they perform in comparison to other known methods on real data. A stripped-down version of the ProxToolbox is used at the University of Göttingen to teach graduate and undergraduate courses in numerical optimization and mathematical imaging. What is omitted from the student version is the repository of algorithms and some of the prox mappings— the students are expected to write these themselves, with some guidance. Experienced researchers, it is expected, will extract the parts of the toolbox they need and incorporate these into their own software. To make it easy to identify the pieces, the toolbox has been organized in a highly modular structure. The modularity comes at the cost of an admittedly labyrinthine structure, which is the hardest thing to master and the main goal of the rest of this tutorial.

6.2 Algorithms

The two different models discussed above, feasibility and constrained optimization (6.19), lead to a natural classification of categories of algorithms. The development presented here follows [25]. To underscore the fact that the algorithms can be applied to problems other than X-ray imaging, the sets involved are denoted by Ω_j for $j = 0, 1, 2, \ldots, m$ and the points of interest are denoted with a u, instead of the

[5] In case you forgot, it's 42.

context-specific notation for a wavefield ψ. The sets Ω_j are subsets of the model space $\mathbb{C}^n$ (or $\mathbb{R}^{2n}$) and, since there can be more than just two sets, as in the case of *phase diversity* or *ptychography* the integer m is just a stand-in for the number of images and other qualitative constraints involved in an experiment.

6.2.1 Model Category I: Multi-set Feasibility

The multi-set feasibility problem is:

Feasibility

$$\text{Find } u \in \bigcap_{j=0}^{m} \Omega_j. \tag{6.32}$$

The numerical experience is that this model format leads to the most effective methods for solving phase-type problems. It is important to keep in mind, however, that for all practical purposes the intersection above is empty, so the algorithm is not really solving the problem since it has no solution.

Feasibility problems can be conveniently reformulated in an optimization format:

$$\min_{u\in\mathbb{C}^n} \sum_{j=0}^{m} \iota_{\Omega_j}(u), \tag{6.33}$$

where ι_{Ω_j} is the indicator function (6.28) of the set Ω_j. The fact that the intersection is empty is reflected in the fact that the optimal value to problem (6.33) is $+\infty$. For the purposes of this tutorial anything bigger than, say, 42 will be approximately infinity.

The easiest iterative procedure of all is the *Cyclic Projections* algorithm

Algorithm 6.2.1
Initialization. *Choose* $u^0 \in (\mathbb{C}^n)$.
General Step *(*$k = 0, 1, \ldots$*)*

$$u^{k+1} \in T_{CP}(u^k) \quad \textit{where} \quad T_{CP} \equiv \mathcal{P}_{\Omega_0}\mathcal{P}_{\Omega_1}\cdots\mathcal{P}_{\Omega_m}.$$

In the context of phase retrieval with one observation and an object domain constraint this is called the *Gerchberg-Saxton* algorithm [26]. An early champion of projection methods for convex feasibility was Censor [27] who together with Cegielski has written a nice review of the extensive literature on these methods [28]. A more recent review of inconsistent feasibility can be found in [29]. The most complete analysis of this algorithm for consistent and inconsistent nonconvex problems has been established in [30] and is reviewed in Chap. 23. In the inconsistent case the fixed points generate cycles of smallest length locally over all other possible cycles generated by projecting onto the sets in the same order. *Rates* of convergence have been established generically for problems with this structure (see [30, Example 3.6]). Rates are important for estimating *how far* a particular iterate is from the solution. The most elementary convergence rate is *linear* convergence, also known as *geometric* or *exponential* convergence in various communities. A sequence $(u^k)_{k=0,1,2,...}$ of points u^k is said to converge linearly (technically, Q-linearly) to a limit point u_* with a global rate $c < 1$ whenever

$$\|u^{k+1} - u_*\| < c\|u^k - u_*\| \qquad \forall k = 0, 1, 2, \dots .$$

The Douglas-Rachford iteration given by (6.15) can only be applied directly to two-set feasibility problems,

$$\text{Find } x \in \Omega_0 \cap \Omega_1 .$$

The fixed point iteration is given by

$$u^{k+1} \in T_{DR}u^k \quad \text{where} \quad T_{DR} \equiv \frac{1}{2}\left(\mathcal{R}_{\Omega_0}\mathcal{R}_{\Omega_1} + \mathrm{Id}\right), \tag{6.34}$$

for $\mathcal{R}_{\Omega_0}$ and $\mathcal{R}_{\Omega_2}$ are generic set reflectors (see (6.13)). It is important not to forget that, even if the feasibility problem is consistent, the fixed points of the Douglas-Rachford Algorithm will not in general be points of intersection. Instead, the *shadows* of the iterates defined as $\mathcal{P}_{\Omega_1}(u^k)$, $k = 0, 1, 2, \dots$, converge to intersection points, when these exist [19].

To extend this to more than two sets, Borwein and Tam [31, 32] proposed the following variant:

Algorithm 6.2.2 (Cyclic Douglas-Rachford—CDR)
Initialization. *Choose* $u^0 \in \mathbb{C}^n$.
General Step $(k = 0, 1, \dots)$

$$u^{k+1} \quad \in \quad T_{CDR}u^k$$
where

$$T_{CDR} \equiv \left(\frac{1}{2}\left(\mathcal{R}_{\Omega_0}\mathcal{R}_{\Omega_1} + \mathrm{Id}\right)\right)\left(\frac{1}{2}\left(\mathcal{R}_{\Omega_1}\mathcal{R}_{\Omega_2} + \mathrm{Id}\right)\right)\cdots \left(\frac{1}{2}\left(\mathcal{R}_{\Omega_m}\mathcal{R}_{\Omega_0} + \mathrm{Id}\right)\right).$$

Different sequencing strategies than the one presented above are possible. In [33] one of the pair of sets is held fixed. This has some theoretical advantages in a convex setting, though no advantage was observed for phase retrieval.

The relaxed Douglas-Rachford algorithm(6.25) takes the general form: for $\lambda \in (0, 1]$

$$(DR\lambda/RAAR) \qquad u^{k+1} \in \left(\frac{\lambda}{2}\left(\mathcal{R}_{\Omega_0}\mathcal{R}_{\Omega_1} + \mathrm{Id}\right) + (1-\lambda)\,\mathcal{P}_{\Omega_1}\right)(u^k). \quad (6.35)$$

Extending this to more than two sets yields the following algorithm, which was first proposed in [25], where it is called CDRλ.

Algorithm 6.2.3 (Cyclic Relaxed Douglas-Rachford CDRλ)
Initialization. *Choose* $u^0 \in \mathbb{C}^n$ *and* $\lambda \in [0, 1]$.
General Step ($k = 0, 1, \ldots$)

$$u^{k+1} \in T_{CDR\lambda}u^k$$

where

$$T_{CDR\lambda} \equiv \left(\frac{\lambda}{2}\left(\mathcal{R}_{\Omega_0}\mathcal{R}_{\Omega_1} + \mathrm{Id}\right) + (1-\lambda)\,\mathcal{P}_{\Omega_1}\right)\cdot \left(\frac{\lambda}{2}\left(\mathcal{R}_{\Omega_1}\mathcal{R}_{\Omega_2} + \mathrm{Id}\right) + (1-\lambda)\,\mathcal{P}_{\Omega_2}\right) \cdots \left(\frac{\lambda}{2}\left(\mathcal{R}_{\Omega_m}\mathcal{R}_{\Omega_0} + \mathrm{Id}\right) + (1-\lambda)\,\mathcal{P}_{\Omega_0}\right).$$

The analysis for RAAR and its precursor, Douglas-Rachford is contained in [30, Sect. 3.2.2] and [34].

Another popular algorithm that can be derived from the Douglas-Rachford algorithm in the convex setting [35] is the Alternating Directions Method of Multipliers (ADMM, [36]). For nonconvex problems like phase retrieval the direct link between these methods is lost, though there have been some recent developments and studies [37–39]. The ADMM algorithm falls into the category of *augmented Lagrangian-based methods*. Here, problem (6.33) is reformulated as

$$\min_{x,u_j\in\mathbb{C}^n}\left\{\iota_{\Omega_0}(x)+\sum_{j=1}^{m}\iota_{\Omega_j}\left(u_j\right)\,\middle|\,u_j=x,\;\; j=1,2,\ldots,m\right\}, \tag{6.36}$$

so that one can apply ADMM to the augmented Lagrangian given by

$$\widetilde{L}_\eta\left(x,u_j,v_j\right)\equiv\iota_{\Omega_0}(x)+\sum_{j=1}^{m}\left(\iota_{\Omega_j}\left(u_j\right)+\left\langle v_j,\; x-u_j\right\rangle+\tfrac{\eta}{2}\|x-u_j\|^2\right), \tag{6.37}$$

where $\eta>0$ is a penalization parameter and v_j, $j=1,2,\ldots,m$, are the multipliers which are associated with the linear constraints. The ADMM algorithm applied to finding the critical points of the corresponding augmented Lagrangian (see (6.37)) is given by

Algorithm 6.2.4 (Nonsmooth ADMM$_1$)
Initialization. *Choose* $x^0, u_j^0, v_j^0\in\mathbb{C}^n$ *and fix* $\eta>0$.
General Step ($k=0,1,\ldots$)

1. *Update*

$$x^{k+1}\in\operatorname{argmin}_{x\in\mathbb{C}^n}\left\{\iota_{\Omega_0}(x)+\sum_{j=1}^{m}\left(\left\langle v_j^k,\; x-u_j^k\right\rangle+\frac{\eta}{2}\|x-u_j^k\|^2\right)\right\}$$
$$=\mathcal{P}_{\Omega_0}\left(\frac{1}{m}\sum_{j=1}^{m}\left(u_j^k-\frac{1}{\eta}v_j^k\right)\right). \tag{6.38}$$

2. *For all* $j=1,2,\ldots,m$ *update (in parallel)*

$$u_j^{k+1}\in\operatorname{argmin}_{u_j\in\mathbb{C}^n}\left\{\iota_{\Omega_j}\left(u_j\right)+\left\langle v_j^k,\; x^{k+1}-u_j\right\rangle+\frac{\eta}{2}\|x^{k+1}-u_j\|^2\right\}$$
$$=\mathcal{P}_{\Omega_j}\left(x^{k+1}-\eta v_j^k\right). \tag{6.39}$$

3. *For all* $j=1,2,\ldots,m$ *update (in parallel)*

$$v_j^{k+1}=v_j^k+\eta\left(x^{k+1}-u_j^{k+1}\right). \tag{6.40}$$

This *can* be written as a fixed point iteration on triplets $(x^k,u^k,v^k)\in\mathbb{C}^n\times\mathbb{C}^{mn}\times\mathbb{C}^n$, but it is not very convenient to see things this way. Note that the projections in Step 2 of the algorithm can be computed in parallel, while the Cyclic Projections and Cyclic Douglas-Rachford Algorithms must be executed sequentially. Note also that the update of the block u^{k+1} incorporates the newest information from the block

x^{k+1} together with the old data v^k, while the update of the block v^{k+1} incorporates the newest information from both blocks x^{k+1} and u^{k+1}. This is in the same spirit as the Gauss-Seidel method for systems of linear equations. Obviously, there is an increase (by a factor of $3 + m$) of the number of variables, but this is a mild increase in complexity in comparison to some recent proposals for phase retrieval which involve *squaring* the number of variables! Indeed ADMM is starting point for just about all the most successful methods for large-scale optimization with linear constraints (see, for instance, [40] and references therein).

An ADMM scheme for phase retrieval has appeared in [41]. This is a terrible algorithm for phase retrieval. It is included here, however, as a point of reference to the Douglas-Rachford Algorithm.

6.2.2 Model Category II: Product Space Formulations

The second category of algorithms is a stepping stone to smooth optimization models, though this is not the most obvious way to motivate the strategy—the connection to smoothing only becomes apparent after some consideration. The idea is to lift the problem to the product space $(\mathbb{C}^n)^{m+1}$ which can be then formulated as a two-set feasibility problem

$$\text{Find } \mathbf{u}^* \in \Omega \cap \mathcal{I},$$

where $\mathbf{u}^* = \left(u_0^*, u_1^*, \ldots, u_m^*\right)$, $\Omega := \Omega_0 \times \Omega_1 \times \cdots \times \Omega_m$ and $\mathcal{I}$ is the diagonal set of $\mathbb{C}^{n(m+1)}$ which is defined by $\{\mathbf{u} = (u, u, \ldots, u) : u \in \mathbb{C}^n\}$. This also involves an increase in the number of unknowns, but only by a factor of m which, while not insignificant when m is large, can be managed through clever implementation. There are two important features of this formulation. First of these is that the projection onto the set Ω can be easily computed since

$$\mathcal{P}_{\Omega}\mathbf{u} = \left(\mathcal{P}_{\Omega_0}u_0, \mathcal{P}_{\Omega_1}u_1, \ldots, \mathcal{P}_{\Omega_m}u_m\right),$$

where $\mathcal{P}_{\Omega_j}$, $j = 1, 2, \ldots, m$, are given in (6.11). The second important feature is that $\mathcal{I}$ is a subspace which also has simple projection given by $\mathcal{P}_{\mathcal{I}}(\mathbf{u}) = \bar{\mathbf{u}}$ where

$$\overline{u}_j = \frac{1}{m+1}\sum_{j=0}^{m} u_j.$$

This formulation immediately suggests the Cyclic Projections algorithm 6.2.1, which, in the case of just two sets, is often called *Alternating Projections*

Algorithm 6.2.5 (Alternating Projections—AP)
Initialization. *Choose* $\mathbf{u}^0 \in \mathbb{C}^{n(m+1)}$.
General Step $(k = 0, 1, \ldots)$

$$\mathbf{u}^{k+1} \in \mathcal{P}_{\mathcal{I}} \mathcal{P}_{\Omega} \mathbf{u}^k.$$

But Algorithm 6.2.5 is equivalent to

$$u_j^{k+1} \in \frac{1}{m+1} \sum_{j=0}^{m} \mathcal{P}_{\Omega_j} u_j^k, \quad j = 0, 1, \ldots, m;$$

in other words, the Alternating Projections algorithm on the product space is equivalent to the Averaged Projections algorithm 6.2.10 and the Alternating Minimization Algorithm 6.2.11. Also the popular *Projected Gradient* method reduces to averaged projections. To see this, consider the following minimization problem:

$$\underset{\mathbf{u} \in \Omega}{\text{minimize}} \ \frac{1}{2} \operatorname{dist}^2(\mathbf{u}, \mathcal{I}). \tag{6.41}$$

The objective above is convex and, because $\mathcal{I}$ is a closed and convex set, continuously differentiable with a Lipschitz continuous gradient given by $\nabla \operatorname{dist}^2(\mathbf{u}, \mathcal{I}) = 2(\mathbf{u} - \mathcal{P}_{\mathcal{I}} \mathbf{u})$ (see (6.46)). The Projected Gradient Algorithm applied to this problem follows easily.

Algorithm 6.2.6 (Projected Gradient—PG)
Initialization. *Choose* $\mathbf{u}^0 \in \mathbb{C}^{n(m+1)}$.
General Step $(k = 0, 1, \ldots)$

$$\begin{aligned} \mathbf{u}^{k+1} \in \mathcal{P}_{\Omega}\left(\mathbf{u}^k - \tfrac{1}{2} \nabla \operatorname{dist}_{\mathcal{I}}^2(\mathbf{u}^k)\right) &\iff \mathbf{u}^{k+1} \in \mathcal{P}_{\Omega} \mathcal{P}_{\mathcal{I}} \mathbf{u}^k \\ &\iff y^{k+1} \in \frac{1}{m+1} \sum_{j=0}^{m} \mathcal{P}_{\Omega_j} y^k, \\ &\qquad \textit{where } u_j^{k+1} = \mathcal{P}_{\Omega_j} y^{k+1}. \end{aligned}$$

This is not surprising since the minimization problem (6.41) is equivalent to (6.48). To see this, note that by the definition of the distance function

$$\min_{\mathbf{u}\in\Omega}\frac{1}{2}\operatorname{dist}^2(\mathbf{u},\mathcal{I})=\min_{\mathbf{u}\in\Omega}\min_{\mathbf{y}\in\mathcal{I}}\frac{1}{2}\|\mathbf{u}-\mathbf{y}\|^2=\min_{\mathbf{u}\in\Omega}\min_{y\in\mathbb{C}^n}\frac{1}{2}\sum_{j=0}^{m}\|y-u_j\|^2$$
$$=\min_{y,\mathbf{u}}\left\{\sum_{j=0}^{m}\frac{1}{2}\|y-u_j\|^2\;\Big|\,u_j\in\Omega_j,\quad j=0,1,\ldots,m\right\}.$$

In the convex setting, the Projected Gradient Algorithm has the advantage that it can be accelerated [42, 43]. A Fast Projected Gradient Algorithm for problem (6.41) looks like:

Algorithm 6.2.7 (Fast Projected Gradient—FPG)
Initialization. *Choose* $\mathbf{u}^0,\mathbf{y}^1\in\mathbb{C}^{n(m+1)}$ *and* $\alpha_k=\frac{k-1}{k+2}$ *for all* $k=0,1,2,\ldots$.
General Step $(k=1,2,\ldots)$

$$\begin{aligned}\mathbf{u}^k&\in\mathcal{P}_\Omega\left(\mathbf{y}^k-\tfrac{1}{2}\nabla\operatorname{dist}^2\left(\mathbf{y}^k,\mathcal{I}\right)\right),\\ \mathbf{y}^{k+1}&=\mathbf{u}^k+\alpha_k\left(\mathbf{u}^k-\mathbf{u}^{k-1}\right)\\ &\Longleftrightarrow\\ \mathbf{u}^k&\in\mathcal{P}_\Omega\mathcal{P}_\mathcal{I}\mathbf{y}^k,\\ \mathbf{y}^{k+1}&=\mathbf{u}^k+\alpha_k\left(\mathbf{u}^k-\mathbf{u}^{k-1}\right).\end{aligned}$$

There is no theory for the choice of acceleration parameter α_k, $k=0,1,2,\ldots$, in Algorithm 6.2.7 for nonconvex problems, but numerical experience [25, 44] indicates that this works pretty well. All that is missing is an explanation.

In the product space setting the best approximation problem takes the form

$$\underset{\mathbf{u}\in\mathbb{C}^{n(m+1)}}{\text{minimize}}\left\{\frac{\lambda}{2(1-\lambda)}\operatorname{dist}^2(\mathbf{u},\mathcal{I})+\iota_\Omega(\mathbf{u})\right\};\tag{6.42}$$

Since there are only two functions, the proximal Douglas-Rachford or the relaxed proximal Douglas-Rachford algorithms apply to (6.42) without any tricks:

Algorithm 6.2.8 *(***Relaxed Douglas-Rachford—DRλ/RAAR***)*
Initialization. *Choose* $\mathbf{u}^0\in\mathbb{C}^{n(m+1)}$ *and* $\lambda\in[0,1]$.
General Step $(k=0,1,\ldots)$

$$\mathbf{u}^{k+1}\in\frac{\lambda}{2}\left(\mathcal{R}_\mathcal{I}\mathcal{R}_\Omega\mathbf{u}^k+\mathbf{u}^k\right)+(1-\lambda)\mathcal{P}_\Omega\mathbf{u}^k.\tag{6.43}$$

The relaxed Douglas-Rachford Algorithm is exactly the proximal Douglas-Rachford algorithm applied to the problem (6.42) [22]; that is,

$$\frac{1}{2}\left(\mathcal{R}_1\mathcal{R}_\Omega + \mathrm{Id}\right) = \frac{\lambda}{2}\left(\mathcal{R}_\mathcal{I}\mathcal{R}_\Omega + \mathrm{Id}\right) + (1-\lambda)\,\mathcal{P}_\Omega,$$

where $\mathcal{R}_1 \equiv 2\,\mathrm{prox}_{1,f_\lambda}(\mathbf{u}) - \mathbf{u}$ is the *proximal reflector* of the function $f_\lambda(\mathbf{u}) \equiv \frac{\lambda}{2(1-\lambda)}\,\mathrm{dist}^2(\mathbf{u}, \mathcal{I})$.

A different kind of relaxation to the Douglas-Rachford algorithm was recently proposed and studied in [45]. This appears to be better than Algorithm 6.2.8. When the sets involved are affine, the algorithm is a convex combination of Douglas-Rachford and Alternating Projections, but generally it takes the form

Algorithm 6.2.9 (Douglas-Rachford-Alternating-Projections)
Initialization. *Choose* $\mathbf{u}^0 \in \mathbb{C}^{n(m+1)}$ *and* $\lambda \in [0, 1]$.
General Step ($k = 0, 1, \ldots$)

$$\mathbf{u}^{k+1} \in \mathcal{P}_\mathcal{I}\left((1+\lambda)\,\mathcal{P}_\Omega\mathbf{u}^k - \lambda\mathbf{u}^k\right) - \lambda\left(\mathcal{P}_\Omega\mathbf{u}^k - \mathbf{u}^k\right). \qquad (6.44)$$

This algorithm is denoted by DRAP in the demonstrations below.

6.2.3 Model Category III: Smooth Nonconvex Optimization

The next algorithm, *Averaged Projections*, could be motivated purely from the feasibility framework detailed above. But there is a more significant *smooth* interpretation of this model, which motivates the smooth model class.

Algorithm 6.2.10 (Averaged Projections—AvP)
Initialization. *Choose* $u^0 \in \mathbb{C}^n$.
General Step ($k = 0, 1, \ldots$)

$$u^{k+1} \in T_{AvP}u^k \quad \textit{where} \quad T_{AvP} \equiv \frac{1}{m+1}\sum_{j=0}^{m}\mathcal{P}_{\Omega_j}.$$

The analysis of averaged projections for problems with this structure is covered by the analysis of nonlinear/nonconvex gradient descent. This is classical and can be found throughout the literature, but it is limited to guarantees of convergence to *critical points* [46, 47]. For phase retrieval it is not known how to guarantee that all critical points are global minimums, though this is a topic of intense interest at the moment.

Although, in general, averaged projections has a slower convergence rate than its sequential counterpart [48], there are two features that recommend this method. First, it can be run in parallel. Secondly, it appears to be more robust to problem inconsistency. Indeed, Averaged Projections algorithm is equivalent to gradient-based schemes when applied to an adequate smooth and nonconvex objective function. This well-known fact goes back to [49] when the sets $\Omega_j,\ j = 0, 1, \ldots, m$, are closed and convex. In particular, two very prevalent schemes are in fact equivalent to AvP.

To see this, consider the problem of minimizing the sum of squared distances to the sets $\Omega_j,\ j = 0, 1, \ldots, m$, that is,

$$\underset{u\in\mathbb{C}^n}{\text{minimize}}\ f(u) \equiv \frac{1}{2(m+1)} \sum_{j=0}^{m} \operatorname{dist}^2\left(u, \Omega_j\right). \tag{6.45}$$

Since the sets $\Omega_j,\ j = 0, 1, \ldots, m$, are nonconvex, the functions $\operatorname{dist}^2\left(u, \Omega_j\right)$ are clearly not differentiable, and hence, same for the objective function $f(u)$. However, in the context of phase retrieval, the sets $\Omega_j,\ j = 0, 1, \ldots, m$, are *prox-regular* (i.e. the projector onto these sets is single-valued near the sets [50]). From elementary properties of prox-regular sets [51] it can be shown that the gradient of the squared distance is defined and differentiable with Lipschitz continuous derivative (that is, the corresponding Hessian) up to the boundary of $\Omega_j,\ j = 0, 1, \ldots, m$, and points where the coordinate elements of the vector u vanish. Indeed, for f given by (6.45)

$$\nabla f(u) \equiv \frac{1}{m+1} \sum_{j=0}^{m} \left(\operatorname{Id} - \mathcal{P}_{\Omega_j}\right)(u). \tag{6.46}$$

Thus, applying the gradient descent with *unit stepsize* to problem (6.45), one immediately recovers the avareged projection algorithm.

The objective in (6.45) is as nice as one could hope for: it has full domain, is smooth and nonegative and has the value zero at points of intersection. These kinds of models are for obvious reasons favored in applications; unfortunately, these reasons are a little old fashioned considering today's mathematical technology for dealing with nonsmooth objectives like (6.33).

Another way to approach problem (6.45) underscores connections with another fundamental algorithmic strategy. Consider the following problem:

$$\min_{u\in\mathbb{C}^n} f(u) \equiv \frac{1}{2} \sum_{j=0}^{m} \operatorname{dist}^2\left(u, \Omega_j\right). \tag{6.47}$$

Using the definition of the function $\operatorname{dist}^2\left(\cdot, \Omega_j\right),\ j = 0, 1, \ldots, m$, problem (6.47) is equivalent to

$$\min_{x,\mathbf{u}} \left\{ \sum_{j=0}^{m} \frac{1}{2} \|x - u_j\|^2 \ \middle|\ u_j \in \Omega_j, \quad j = 0, 1, \ldots, m \right\}, \tag{6.48}$$

where $\mathbf{u} = (u_0, u_1, \ldots, u_m) \in (\mathbb{C}^n)^{m+1}$. The number of variables has now increased $(m+1)$-fold, which, for applications like ptychography starts to get worrying since m can be large. But this is more a conceptual issue than practical.

Problem (6.48) always has an optimal solution (the objective is continuous and the constraint is closed and bounded set, so by a theorem from Weierstrass the minimum is attained). The optimization problem (6.48) consists of constraint sets which are separable over the variables u_j, $j = 0, 1, \ldots, m$; this can be exploited to divide the optimization problem into a sequence of easier subproblems. Alternating Minimization (AM) does just this, and involves updating each variable sequentially:

Algorithm 6.2.11 (Alternating Minimization—AM)
Initialization. *Choose* $\left(y^0, u_0^0, u_1^0, \ldots, u_m^0\right) \in (\mathbb{C}^n)^{m+2}$.
General Step $(k = 0, 1, \ldots)$

1. *Update*

$$y^{k+1} = \operatorname{argmin}_{y \in \mathbb{C}^n} \sum_{j=0}^{m} \frac{1}{2}\|y - u_j^k\|^2 = \frac{1}{m+1}\sum_{j=0}^{m} u_j^k. \tag{6.49}$$

2. *For all* $j = 0, 1, \ldots, m$ *update (in parallel)*

$$u_j^{k+1} \in \operatorname{argmin}_{u_j \in \Omega_j} \frac{1}{2}\|u_j - y^{k+1}\|^2 = \mathcal{P}_{\Omega_j} y^{k+1}. \tag{6.50}$$

By combining (6.49) and (6.50), the algorithm is written compactly as

$$y^{k+1} \in \frac{1}{m+1}\sum_{j=0}^{m} \mathcal{P}_{\Omega_j} y^k, \tag{6.51}$$

which is just averaged projections, Algorithm 6.2.10! When $m = 1$, i.e., only one image is considered, the Alternating Minimization Algorithm above coincides with what is known as the *Error Reduction* Algorithm [52] in the optics community.

In [53] Marchesini studied an *augmented Lagrangian* approach to solving

$$\min_{\psi \in \mathbb{C}^n} \frac{1}{2n} \sum_{j=0}^{m}\sum_{i=1}^{n} \left(\left\|\left(\mathcal{D}_j(\psi)\right)_i\right\| - \sqrt{I_{ij}}\right)^2. \tag{6.52}$$

This is a nonsmooth least-squares relaxation of problem (6.2). Generic linear least squares problems take the form

$$\min_{y \in \mathbb{C}^n} \frac{1}{2n} \sum_{j=0}^{m}\sum_{i=1}^{n} \left(\left\|\left(F_j(y)\right)_i\right\| - b_{ij}\right)^2 \tag{6.53}$$

where F_j is a linear mapping from $\mathbb{C}^n$ to $\mathbb{C}^n$ and b_{ij} are positive scalars. When the sets Ω_j are defined by

$$\Omega_j \equiv \left\{u \in \mathbb{C}^n \mid \left\|\left(F_j(u)\right)_i\right\| = b_{ij}\right\} \tag{6.54}$$

and the variables $y \in \mathbb{C}^n$ and $\mathbf{u} = (u_1, u_2 \ldots, u_m) \in \mathbb{C}^{mn}$ with $u_j \in \mathbb{C}^n$ satisfy $y = u_j$ for each $j = 1, 2, \ldots, m$, the resulting primal-dual/ADMM Algorithm takes the form:

Algorithm 6.2.12 (AvP^2)
Initialization. *Choose any* $x^0 \in \mathbb{C}^n$ *and* $\rho_j > 0,\ j = 0, 1, \ldots, m$. *Compute* $u_j^1 \in \mathcal{P}_{\Omega_j}\left(y^0\right)$ $(j = 0, 1, \ldots, m)$ *and* $y^1 \equiv (1/\,(m+1)) \sum_{j=0}^{m} u_j^1$.
General Step. *For each* $k = 1, 2, \ldots$ *generate the sequence* $\left\{\left(y^k, \mathbf{u}^k\right)\right\}_{k=0,1,2,\ldots}$ *as follows:*

- *Compute*

$$y^{k+1} = \frac{1}{m}\sum_{j=1}^{m}\left(u_j^k + \frac{1}{\rho_j}\left(y^k - y^{k-1}\right)\right). \tag{6.55}$$

- *For each* $j = 1, 2, \ldots, m$, *compute*

$$u_j^{k+1} = \mathcal{P}_{\Omega_j}\left(u_j^k + \frac{1}{\rho_j}\left(2y^k - y^{k-1}\right)\right). \tag{6.56}$$

This algorithm can be viewed as a smoothed/relaxed version of Algorithm 6.2.4, but, when you look at it for the first time, the most obvious thing that jumps out at you is that this is averaged projections with a tw-step recursion. This is why it has been called AvP^2 in [25].

The more general PHeBIE Algorithm 6.2.13 applied to the problem of blind ptychography [54] reduces to Averaged Projections Algorithm for phase retrieval when the illuminating field is known. To derive this method, note that for any fixed y and $\mathbf{u}$, the function $u \mapsto \mathbf{F}(z, y, \mathbf{u})$ given by (6.21) is continuously differentiable and its partial gradient, $\nabla_z \mathbf{F}(z, y, \mathbf{u})$, is Lipschitz continuous with moduli $L_z(y, \mathbf{u})$. The same assumption holds for the function $y \mapsto \mathbf{F}(z, y, \mathbf{u})$ when z and $\mathbf{u}$ are fixed. In this case, the Lipschitz moduli is denoted by $L_y(z, \mathbf{u})$. Define $L'_z(y, \mathbf{u}) \equiv \max\{L_z(y, \mathbf{u}), \eta_z\}$ where η_z is an arbitrary positive number. Similarly define $L'_y(z, \mathbf{u}) \equiv \max\left\{L_y(z, \mathbf{u}), \eta_y\right\}$ where η_y is an arbitrary positive number. The constant η_z and η_y are used to address the following issue: if the Lipschitz constants $L_z(y, \mathbf{u})$ and/or $L_y(z, \mathbf{u})$ are zero then one should replace them with positive numbers (for the sake of well-definedness of the algorithm). In practice, it is better to chose them to be small numbers but for the analysis it can be chosen arbitrarily.

Algorithm 6.2.13 (Proximal Heterogeneous Block Implicit-Explicit)
Initialization. *Choose* $\alpha, \beta > 1$, $\gamma > 0$ *and* $\left(z^0, y^0, \mathbf{u}^0\right) \in X \times O \times M$.
General Step $(k = 0, 1, \ldots)$

1. *Set* $\alpha^k = \alpha L'_z\left(y^k, \mathbf{u}^k\right)$ *and select*

$$z^{k+1} \in \operatorname{argmin}_{z \in X} \left\{ \left\langle z - z^k,\ \nabla_z \mathbf{F}\left(z^k, y^k, \mathbf{u}^k\right)\right\rangle + \frac{\alpha^k}{2}\|z - z^k\|^2 \right\}, \tag{6.57}$$

2. *Set* $\beta^k = \beta L'_y\left(z^{k+1}, \mathbf{u}^k\right)$ *and select*

$$y^{k+1} \in \operatorname{argmin}_{y \in O} \left\{ \left\langle y - y^k,\ \nabla_y \mathbf{F}\left(z^{k+1}, y^k, \mathbf{u}^k\right)\right\rangle + \frac{\beta^k}{2}\|y - y^k\|^2 \right\}, \tag{6.58}$$

3. *Select*

$$\mathbf{u}^{k+1} \in \operatorname{argmin}_{\mathbf{u} \in M} \left\{ \mathbf{F}\left(z^{k+1}, y^{k+1}, \mathbf{u}\right) + \frac{\gamma}{2}\|\mathbf{u} - \mathbf{u}^k\|^2 \right\}. \tag{6.59}$$

Algorithm 6.2.13, referred to as PHeBIE in Sect. 6.3.3, can be interpreted as a combination of the algorithm proposed in [55] and a slight generalization of the PALM Algorithm [47]. In the context of blind ptychography (6.4), the block of variables y is replaced with the object ψ and the function $\mathbf{F}$ is the least squares objective (6.21). A partially preconditioned version of PALM was studied in [56] for phase retrieval, with improved performance over PALM. The regularization parameters α^k and β^k, $k = 0, 1, 2, \ldots$, are discussed in [54]. These parameters are inversely proportional to the step size in Steps (6.57) and (6.58) of the algorithm. Noting that α_k and β_k, $k = 0, 1, 2, \ldots$, are directly proportional to the respective partial Lipschitz moduli, the larger the partial Lipschitz moduli the *smaller* the step size, and hence the slower the algorithm progresses.

This brings to light an advantage of blocking strategies that is discussed in Chap. 12: algorithms that exploit block structures inherent in the objective function achieve better numerical performance by taking heterogeneous step sizes optimized for the separate blocks. There is, however, a price to be paid in the blocking strategies that are explored here: namely, they result in procedures that pass *sequentially* between operations on the blocks, and as such are not immediately parallelizable. The ptychography application is very generous in that it permits parallel computations on highly segmented blocks.

Nonsmooth analysis can be applied to the objective in (6.53), but this is still not main stream enough to be the stuff of normal graduate training, so it remains rather exotic. A popular way around this, is to formulate (6.53) as a system of quadratic equations:

$$\left\| \left(F_j(u)\right)_i \right\|^2 = b_{ij}^2, \quad \forall\ j = 1, 2, \ldots, m, \ \forall\ i = 1, 2, \ldots, n. \tag{6.60}$$

The corresponding squared least squares residual of the quadratic model is plenty smooth:

$$\min_{u\in\mathbb{C}^n} G(u) \equiv \frac{1}{2}\sum_{j=0}^{m}\sum_{i=1}^{n}\left(\left\|\left(F_j(u)\right)_i\right\|^2 - b_{ij}^2\right)^2. \tag{6.61}$$

There is a trick from conic programming that allows you to recast a quadratic equation on $\mathbb{R}^n$ as a linear equation on the space of matrices, $\mathbb{R}^{n\times n}$. The idea in the context of phase retrieval is called *phase lift*. The problem here is that, though the quadratic equation has been replaced by a linear equation (albeit in a much larger space), the desired solution is rank 1. Even though the set of fixed rank matrices is a manifold, it is not convex, so there is conservation of difficulty. The way around this is to replace the rank constraint with a norm—otherwise known as *convex relaxation*. The reasoning here is that, for convex problems, all local solutions are also global solutions to the problem; so you solve your convex problem and—poof— you have the global solution to the nonconvex problem under certain (hard to verify) conditions that guarantee the correspondence of the two problems. This is attractive as an analytical strategy, but as an algorithmic strategy it is not practical. Blumensath and Davies [57, 58] were the first ones to ask the question whether the conditions that guarantee correspondence of the nonconvex problem and its convex relaxation are also sufficient to guarantee that the nonconvex problem doesn't have any critical points *other than* global minima. They answered this question in the affirmative for the projected gradient Algorithm 6.2.6 and Hesse, Luke and Neumann showed that this is also the case for alternating projections Algorithm 6.2.5 [59]. So, there is no need to resort to convex relaxations, which is good news indeed since the phase lift method is not implementable on standard consumer-grade architectures for any of the Göttingen data sets.
Other methods based on the quartic objective have gained popularity in the newer generation of phase retrieval studies in the applied mathematics community. Notable among these are methods called Wirtinger flow. Smoothness makes the analysis easier, but the quartic objective has almost no curvature around critical points, which makes convergence of first order methods much slower than first order methods applied to nonsmooth objectives. See [14, Sect. 5.2] for a discussion of this and [25] for numerical comparisons.

Accelerations. The formulation of problem (6.45) has a fixed weight between the various distances. An extension of this is a dynamically weighted average between the projections to the sets Ω_j, $j = 0, 1, \ldots, m$. This idea was proposed in [14] where it is called *extended least squares*. In the context of sensor localization a similar approach was also proposed in [60] where it is called Sequential Weighted Least Squares (SWLS). The underlying model in [14] is the negative log-likelihood measure of the sum of squared set distances:

$$\underset{u\in\mathbb{C}^n}{\text{minimize}} \sum_{j=0}^{m} \ln\left(\text{dist}^2\left(u, \Omega_j\right) + c\right), \qquad (c > 0). \tag{6.62}$$

Gradient descent applied to this objective yields what was called the *Dynamically Reweighted Averaged Projections* Algorithm (DyRePr) in [25].

Algorithm 6.2.14 (Dynamically Reweighted Averaged Projections)
Initialization. *Choose* $u^0 \in \mathbb{C}^n$ *and* $c > 0$.
General Step ($k = 0, 1, \ldots$)

$$u^{k+1} \in u^k - \sum_{j=0}^{m} \frac{2}{\left(\operatorname{dist}^2\left(u^k, \Omega_j\right) + c\right)} \left(u^k - \mathcal{P}_{\Omega_j} u^k\right). \tag{6.63}$$

The smoothness of the sum of squared distances (almost everywhere) opens the door to higher-order techniques from nonlinear optimization that accelerate the basic gradient descent method. Quasi-Newton methods, for instance, would do the trick, and as observed in [61], they work unexpectedly well even on nonsmooth problems.

Algorithm 6.2.15 (Limited Memory BFGS with Trust Region)

1. *(Initialization) Choose* $\tilde{\eta} > 0$, $\zeta > 0$, $\overline{\ell} \in \{1, 2, \ldots, n\}$, $u^0 \in \mathbb{C}^n$, *and set* $\nu = \ell = 0$. *Compute* $\nabla f\left(u^0\right)$ *and* $\|\nabla f\left(u^0\right)\|$ *for*

$$f(u) \equiv \frac{1}{2(m+1)} \sum_{j=0}^{m} \operatorname{dist}^2\left(u, \Omega_j\right), \quad \nabla f(u) \equiv \frac{1}{m+1} \sum_{j=0}^{m} \left(\operatorname{Id} - \mathcal{P}_{\Omega_j}\right)(u).$$

2. *(L-BFGS step) For each* $k = 0, 1, 2, \ldots$ *if* $\ell = 0$ *compute* u^{k+1} *by some line search algorithm; otherwise compute*

$$s^k = -\left(M^k\right)^{-1} \nabla f\left(u^k\right),$$

where M^k *is the L-BFGS update [62],* $u^{k+1} = u^k + s^k$, $f\left(u^{k+1}\right)$, *and the* predicted change *(see, for instance [63]).*

3. *(Trust Region) If* $\rho\left(s^k\right) < \tilde{\eta}$, *where*

$$\rho\left(s^k\right) = \frac{\textit{actual change at step } k}{\textit{predicted change at step } k},$$

reduce the trust region Δ^k, *solve the trust region subproblem for a new step* s^k *[64], and return to the beginning of Step 2. If* $\rho\left(s^k\right) \geq \tilde{\eta}$ *compute* $u^{k+1} = u^k + s^k$ *and* $f\left(u^{k+1}\right)$.

4. (Update) Compute $\nabla f\left(u^{k+1}\right)$, $\|\nabla f\left(u^{k+1}\right)\|$,

$$y^k \equiv \nabla f\left(u^{k+1}\right) - \nabla f\left(u^k\right), \quad s^k \equiv u^{k+1} - u^k,$$

and ${s^k}^T y^k$. *If* ${s^k}^T y^k \leq \zeta$, *discard the vector pair* $\{s^{k-\ell}, y^{k-\ell}\}$ *from storage, set* $\ell = \max\{\ell - 1, 0\}$, $\Delta^{k+1} = \infty$, $\mu^{k+1} = \mu^k$ *and* $M^{k+1} = M^k$ *(i.e. shrink the memory and don't update); otherwise set* $\mu^{k+1} = \frac{{y^k}^T y^k}{{s^k}^T y^k}$ *and* $\Delta^{k+1} = \infty$, *add the vector pair* $\{s^k, y^k\}$ *to storage, if* $\ell = \overline{\ell}$, *discard the vector pair* $\{s^{k-\ell}, y^{k-\ell}\}$ *from storage. Update the Hessian approximation* M^{k+1} *[62]. Set* $\ell = \min\{\ell + 1, \overline{\ell}\}$, $\nu = \nu + 1$ *and return to Step 1.*

This looks complicated but is standard in nonlinear optimization. Convergence is still unexplained for the limited memory implementation.

6.3 ProxToolbox—A Platform for Creative Hacking

A platform for collecting and working with data should satisfy several objectives:

- data transfer
- sharing data processing algorithms
- comparing the performance of different algorithmic approaches
- teaching
- innovation.

The ProxToolbox has been used within the Collaborative Research Center Nanoscale Photonic Imaging (SFB 755) at the University of Göttingen for each of the points above. It is written to be able to incorporate new problems, data, and algorithms without abandoning the old knowledge. This type of built-in knowledge retention requires a structure that is burdensome for single-purpose users. Most colleagues and students prefer to cannibalize the ProxToolbox—hacking is positively encouraged. This tutorial and the demos in the toolbox are intended to put the user on a fast track to successfully disassembling and re-purposing the basic elements.

Our presentation of the toolbox here is without specific reference to commands and code to prevent this tutorial from being outdated within a few months. Certain aspects of the code will change as new applications and new features get added to the toolbox, but what will not change is the compartmentalization of various mathematically and computationally distinct tools.

To download the toolbox and the data go to

```
http://num.math.uni-goettingen.de/proxtoolbox/
```

Here you will find links to the Matlab and Python versions of the toolbox, along with documentation and literature. The toolbox has the following organizational structure:

- Nanoscale_Photonic_Imaging_demos
- Algorithms
- ProxOperators
- Drivers/Problems
  - ...
  - Phase
    - Demos
    - DataProcessors
    - ProxOperators
  - Ptychography
    - Demos
    - DataProcessors
    - ProxOperators
  - ...
- Utilities
- InputData
- Documentation

The *Nanoscale_Photonic_Imaging_demos folder*. This folder contains scripts to generate the figures shown in this tutorial. This is the rabbit you will follow down the hole.

The *Algorithms* folder. This folder contains a general *algorithm wrapper* that loops through the iterations calling the desired algorithm. This is exactly T in (6.26), and the T_* indicates which specific algorithm is run, from Algorithm 6.2.1 through Algorithm 6.2.9. After the specific fixed point operator is applied, a specialized *iterate monitor* is called. This will depend both on the problem and the algorithm being run. The default is a *generic iterate monitor* that merely checks the distance between successive iterates. By default, the stopping criterion for the fixed point iteration is when the step between successive iterates falls below a tolerance given by the user. But for some algorithms and some problems, this may not be the best or most informative data about the progress of the iteration. For instance, if the problem is a feasibility problem (6.32), then the *feasibility iterate monitor* not only computes the difference between successive iterates, but also the distance between sets (the gap) at a given iterate. In this context, a reasonable comparison between algorithms is not the step-size, but rather between the gap achieved by different algorithms. If one is running a Douglas-Rachford-type algorithm on a feasibility problem, then as explained above, the iterates themselves don't have to converge, but their *shadows*, defined as the projection of the iterates onto one of the sets, will give a good indication of convergence of some form. Still other algorithms, like ADMM 6.2.12, generate several sequences of iterates (three in the case of ADMM), only two of which converge nicely when everything goes well [65]. As much as possible, the iterate monitoring is automated so that the user does not have to bother with this. But users who are interested in algorithm development will want to pay close attention to this.

The *ProxOperators* folder. Some prox operators are generic, like a projector onto the diagonal of a product space $\mathcal{I}$ (otherwise known as averaging), or the prox of the ℓ_1-norm (soft thresholding), or the prox of the ℓ_0-function (hard thresholding). General prox operators are stored here. These always map an input to another point in the same space, but how they do this depends on the strucure of the input u (array, dimension, etc.). Some problems involve prox mappings that are specific to that problem, like Sudoku. These specific prox mappings are stored under the Problem/Drivers folder.

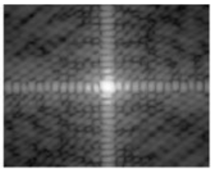
The *Drivers/Problems folder*. This is the folder where the specific problem instances are stored. The problems that are of interest for this chapter are phase and ptychography, though there are other problems, like computed tomography, sensor localization and Sudoku. The *Phase* subfolder contains a general problem family handler called "phase". Since all phase problems have similar features, this problem handler makes sure that all the inputs and outputs are processed in the same way. The toolbox works through input files stored in the *demos* subfolder. The input files contain names of data sets, data processors, algorithm names and parameters, and other user defined parameters like stopping tolerances, output choices and so forth. The input files might be augmented by a graphical user interface in the future. The link between the experimentalist and the mathematician is through the *data processor*. The data processor for the Göttingen data sets is easily identified and contains all the required parameter values for specific experiments conducted at the Institute for Physics in Göttingen. The data that the processor manipulates is not contained in the ProxToolbox release, but is stored separately and must be downloaded from the links provided on the ProxToolbox homepage. Prox operators specific to phase retrieval, such as the projection onto the intensity data (6.11), are also stored at this level.
The *InputData folder*. The data, which is intended to be stored or linked in the directory "InputData", is not included in the software toolbox in the interest of portability. This tutorial will only cover demonstrations with the *Phase* datasets and the *Ptychography* datasets. As these sets grow and develop, the links may change to reflect different hosts.

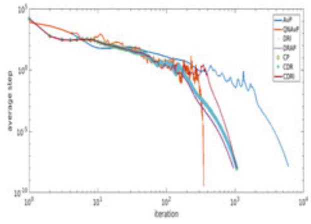
The *Utilities folder*. This is where generic image and data manipulation tools are stored.

6.3.1 Coffee Break

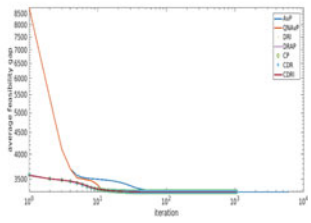
The first walk through the Toolbox is demonstrated on an image set produced by undergraduates in Tim Salditt's laboratory at the Institute for X-ray Physics at the University of Göttingen. The data is the *CDI* intensity datafile contained in the *Phase* dataset linked to the ProxToolbox homepage. There is a demonstration of the Cyclic Projections Algorithm 6.2.1 in the folder *Nanoscale_Photonic_Imaging_demos*. To run the demonstration, just type *Coffee_demo* at the Matlab prompt (assuming Matlab has the demo folder and all the data folders in its path) or *python Coffee_demo.py* at the shell prompt if you are working with Python. The data set presented here

(I_i, in problem (6.2) with $n = 128^2$) is a diffraction image of visible light shown in Fig. 6.1a (log intensity scale). The physical parameters of the image (magnification factor, Fresnel number, etc.) are not given, so the easiest thing to do is to assume a perfect imaging system and expert experimentalists. The imaging model is then just an unmodified Fourier transform, that is, far-field imaging. The object was a real, nonnegative obstacle, supported on some patch in the object plane, so that the qualitative constraint O is of the form (6.23). The only way to know that the algorithm is converging at least to a local best approximation point is to monitor the successive iterates and feasibility gap Fig. 6.1b–c. A small feasibility gap is not necessarily desireable, since this also means that the noise is being faithfully recovered. For the demonstration shown here, a low-pass filter is applied to the data since almost all of the recoverable information about the object is contained in the low-frequency elements. It might seem counterintuitive, but the larger the feasibility gap (i.e., the more *inconsistent the problem is*), the faster the algorithm converges. In Chap. 23 this is explained. The original object was a coffee cup which the generous reader can see if he tilts his head to the left and squints really hard (Fig. 6.1d right). When you run the demo, don't be surprised if your reconstruction is an upside down version of what is shown here—this is a symptom of nonuniqueness of solutions to the phase problem.


(a)



(b)



(c)

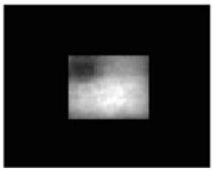

(d)

Fig. 6.1 **a** Observation (log scale) from optical diffraction experiment. **b** Step-size and **c** gap size between constraint sets versus iteraton for several algorithms [25]. **d** Typical recovery from the algorithms

6.3.2 Star Power

The next demonstration is of the reconstruction of a test object (the Siemens star) from near field X-ray data provided by Tim Salditt's laboratory at the Institute for X-ray Physics at the University of Göttingen. Here the structured illumination shown in Fig. 6.2a left is modeled by $\mathcal{D}_j$, $j = 1, 2, \ldots, m$, in problem (6.3) with $m = 1$. The image shown in Fig. 6.2a right is in the near field, so the mapping $\mathcal{D}_F$ in problem (6.2) is the near-field *Fresnel transform* [66]. In the model (6.3) this image is represented by I_{ij}, $j = 1, 2, \ldots, m$, with $m = 1$. The qualitative constraint is that the object is a *pure phase object*, that is, the field in the object domain has amplitude 1 at each pixel.

A reconstruction with this data that does not take noise into account is shown in Fig. 6.3. What is remarkable here is that if one only looks at the convergence of the algorithms and judges by the achieved gap before termination, it appears that the quasi-Newton accelerated average projections algorithm (QNAvP) is clearly the best Fig. 6.3b–c. But when you look at the reconstructions Fig. 6.3a the QNAvP reconstruction is the worst. The problem here is that the noise has also been faithfully recovered.

In [67] a regularization strategy is proposed that blows a ball around the data (either Euclidean or Kullback-Leibler, as appropriate) and takes any reasonable point within the ball. The justification is that, if the data is noisy anyway, you don't want to match

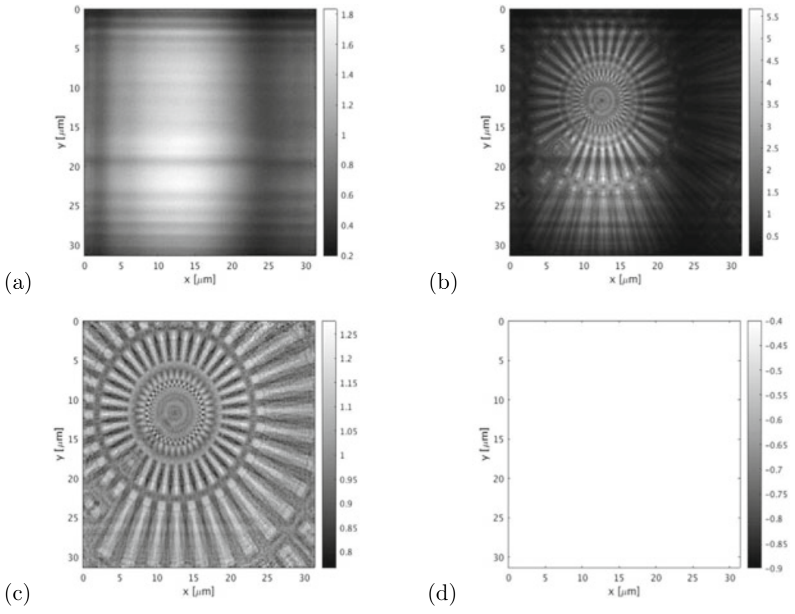
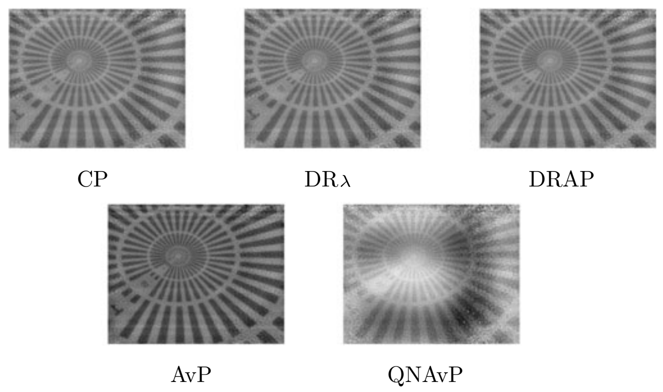


Fig. 6.2 Near field X-ray holography experiment with a Siemens test object [25]. **a** The empty beam. **b** Observed pattern. **c** Initial guess for object amplitude. **d** Initial guess for object phase

(a)



(b)

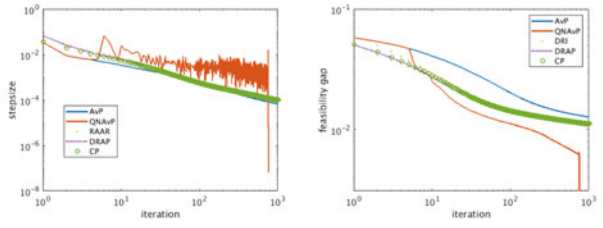


Fig. 6.3 **a** Reconstruction of regularized near field holography experiment with empty beam correction for the same data shown in Fig. 6.2. **b** Step-size and **c** gap between the constraint sets versus iteration [25]

it exactly. For a more precise development of this intuition see Chap. 4. Though it is not obvious, projecting onto such fattened data sets is more expensive than projecting onto the original noisy data. The latter is viewed as an approximate and extrapolated projection onto the fattened set. The algorithm is terminated when the iterates are within a tolerable distance of the data. A demonstration of this is shown in Fig. 6.4. The theoretical justification for this strategy is quite technical, but effectively what one is doing is running the old algorithms with early termination.

6.3.3 E Pluribus Unum

The demonstration *Ptychography_demo* shown in Fig. 6.5 computes the probe and object from a far-field raster scan of 676 overlapping patches of the Siemens star, illuminated by a narrow X-ray beam. The mathematical problem is to minimize the objective function given in (6.3). The demonstration shows how the PHeBIE algorithm 6.2.13 does this. Since blind deconvolution has many local solutions, the process has two phases: the first conventional phase retrieval on the data with a probe

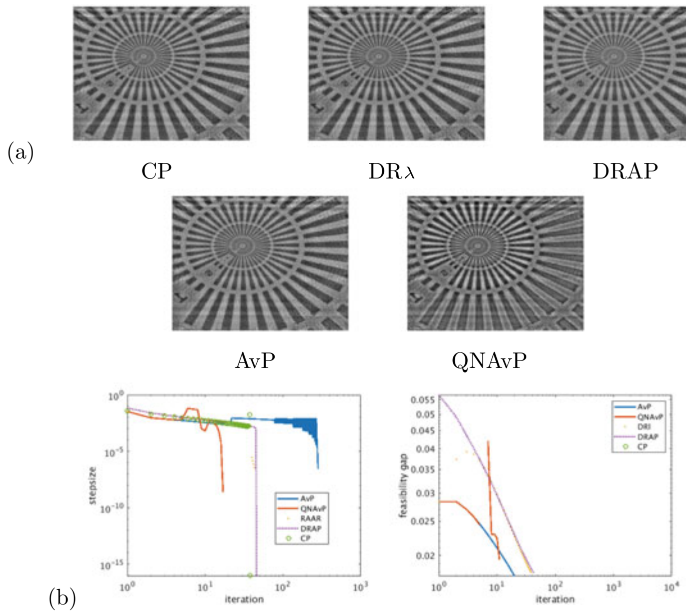


Fig. 6.4 **a** Reconstruction of *regularized* (i.e. early termination) near field holography experiment with empty beam correction for the same data shown in Fig. 6.2a. **b** Step-size and **c** gap between the constraint sets versus iteration

ansatz, the second phase simultaneous phase retrieval and probe determination in what is essentially a nonlinear blind deconvolution. In the *Ptychography_demo* the first phase is executed with the DRλ algorithm on the product space: 676 images of size (192^2) accounting for 26 equal translations of the beam side to side and 26 equal shifts of the beam top to bottom. The second phase is executed with the PHeBIE algorithm starting from the last iterate of the first phase.

6.4 Last Word

One of the most rewarding things about participating in the Nanoscale Photonic Imaging Collaborative Research Center at the University of Göttingen has been working with scientists from different disciplines with different sensibilities and intuition. Collaboration starts with mutual respect and an openness for new ways of thinking about things. This has resulted in better mathematics and better science, both grounded in real world experience but with an attention to abstract structures. This has forced the examination of aspects of abstract models that, at first glance, don't seem that important, but turn out to be decisive in practice.

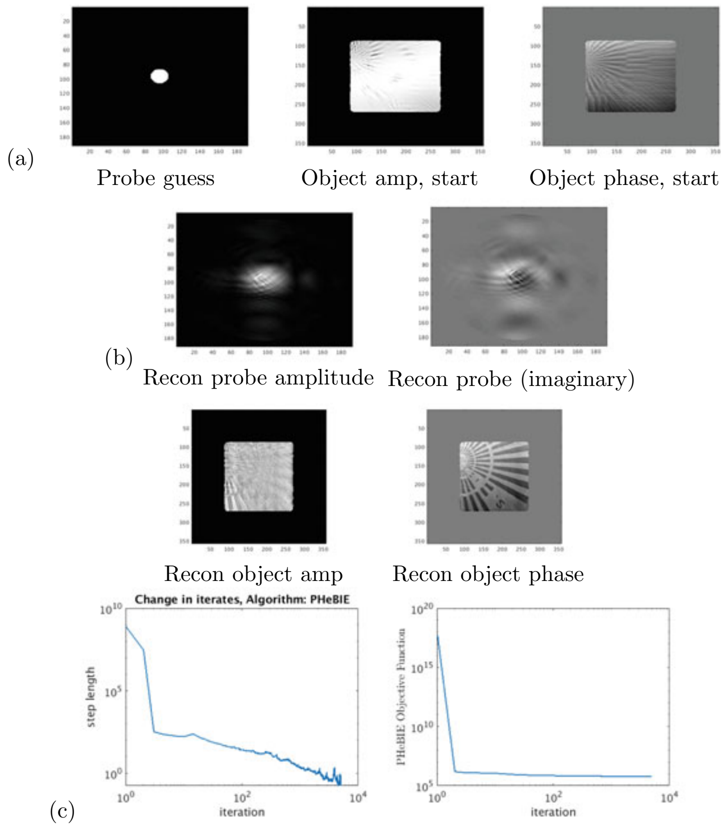


Fig. 6.5 **a** Initial probe and warm start object initialization of a far field ptychography experiment for the scan data. **b** Probe and object reconstructed by the PHeBIE algorithm 6.2.13. **c** Step size and objective function values versus iteration

Acknowledgements This work was supported by DFG grant SFB755.

References

1. Krantz, S.G.: Mathematical Apocrypha Redux. Mathematical Association of America, UK edition (2006)
2. Courant, R., Hilbert, D.: Methods of Mathematical Physics. Interscience Publishers, New York (1953)
3. Luke, D.R.: Proxtoolbox. http://num.math.uni-goettingen.de/proxtoolbox/ (2017)
4. Hegerl, R., Hoppe, W.: Dynamische Theorie der Kristallstrukturanalyse durch Elektronenbeugung im inhomogenen Primärstrahlwellenfeld. Ber. Bunsenges. Phys. Chem **74**(11), 1148–1154 (1970)
5. Maiden, A.M., Rodenburg, J.M.: An improved ptychographical phase retrieval algorithm for diffractiveimaging. Ultramicroscopy **109**, 1256–1262 (2009)

6. Rodenburg, J.M., Bates, R.H.T.: The theory of super-resolution electron microscopy via Wigner-distribution deconvolution. Philos. Trans. R. Soc. Lond. Ser. A **339**, 521–553 (1992)
7. Rodenburg, J.M., Hurst, A.C., Cullis, A.G., Dobson, B.R., Pfeiffer, F., Bunk, O., David, C., Jefimovs, K., Johnson, I.: Hard-X-ray lensless imaging of extended objects. Phys. Rev. Lett. **98**, 034801 (2007)
8. Thibault, P., Dierolf, M., Bunk, O., Menzel, A., Pfeiffer, F.: Probe retrieval in ptychographic coherent diffractive imaging. Ultramicroscopy **109**, 338–343 (2009)
9. Robisch, A.L., Salditt, T.: Phase retrieval for object and probe using a series of defocus near-field images. Opt. Express **21**(20), 23345–23357 (2013)
10. Gonsalves, R.A.: Phase retrieval and diversity in adaptive optics. Opt. Eng. **21**(5), 829–832 (1982)
11. Robisch, A.L., Kröger, K., Rack, A., Salditt, T.: Near-field ptychography using lateral and longitudinal shifts. New J. Phys. **17**(7), 073033 (2015)
12. Robisch, A.L.: Phase retrieval for object and probe in the optical near-field. Ph.D. thesis, Universität Göttingen (2016)
13. Fienup, J.R.: Phase retrieval algorithms: a comparison. Appl. Opt. **21**(15), 2758–2769 (1982)
14. Luke, D.R., Burke, J.V., Lyon, R.G.: Optical wavefront reconstruction: Theory and numerical methods. SIAM Rev. **44**, 169–224 (2002)
15. Bauschke, H.H., Combettes, P.L., Luke, D.R.: A hybrid projection reflection method for phase retrieval. J. Opt. Soc. Amer. A **20**(6), 1025–1034 (2003)
16. Bauschke, H.H., Combettes, P.L., Luke, D.R.: Phase retrieval, error reduction algorithm and Fienup variants: a view from convex feasibility. J. Opt. Soc. Amer. A **19**(7), 1334–1345 (2002)
17. Douglas Jr., J., Rachford Jr., H.H.: On the numerical solution of heat conduction problems in two or three space variables. Trans. Amer. Math. Soc. **82**, 421–439 (1956)
18. Lions, P.L., Mercier, B.: Splitting algorithms for the sum of two nonlinear operators. SIAM J. Numer. Anal. **16**, 964–979 (1979)
19. Bauschke, H.H., Combettes, P.L., Luke, D.R.: Finding best approximation pairs relative to two closed convex sets in Hilbert spaces. J. Approx. Theory **127**, 178–192 (2004)
20. Miao, J., Charalambous, P., Kirz, J., Sayre, D.: Extending the methodology of X-ray crystallography to allow imaging of micrometre-sized non-crystalline specimens. Nature **400**, 342–344 (1999)
21. Luke, D.R.: Relaxed averaged alternating reflections for diffraction imaging. Inverse Probl. **21**, 37–50 (2005)
22. Luke, D.R.: Finding best approximation pairs relative to a convex and a prox-regular set in Hilbert space. SIAM J. Optim. **19**(2), 714–739 (2008)
23. Moreau, J.J.: Fonctions convexes duales et points proximaux dans un espace Hilbertien. Comptes Rendus de l'Académie des Sciences de Paris **255**, 2897–2899 (1962)
24. Adams, D.: The Hitchhikers Guide to the Galaxy. Pan Books, New York (1980)
25. Luke, D.R., Sabach, S., Teboulle, M.: Optimization on spheres: models and proximal algorithms with computational performance comparisons. SIAM J. Math. Data Sci. **1**(3), 408–445 (2019)
26. Gerchberg, R.W., Saxton, W.O.: A practical algorithm for the determination of phase from image and diffraction plane pictures. Optik **35**, 237–246 (1972)
27. Censor, Y.: Row-action methods for huge and sparse systems and their applications. SIAM Rev. **23**, 444–466 (1981)
28. Censor, Y., Cegielski, A.: Projection methods: an annotated bibliography of books and reviews. Optimization **64**, 2343–2358 (2015). https://doi.org/10.1080/02331934.2014.957701
29. Censor, Y., Zaknoon, M.: Algorithms and convergence results of projection methods for inconsistent feasibility problems: a review. Pure Appl. Funct. Anal. (2019). https://arxiv.org/abs/1802.07529 (to appear)
30. Luke, D.R., Thao, N.H., Tam, M.K.: Quantitative convergence analysis of iterated expansive, set-valued mappings. Math. Oper. Res. **43**(4), 1143–1176 (2018). https://doi.org/10.1287/moor.2017.0898
31. Borwein, J.M., Tam, M.K.: A cyclic Douglas-Rachford iteration scheme. J. Optim. Theory Appl. **160**(1), 1–29 (2014)

32. Borwein, J.M., Tam, M.K.: The cyclic Douglas-Rachford method for inconsistent feasibility problems. J. Nonlinear Convex Anal. **16**(4), 537–584 (2015)
33. Bauschke, H.H., Noll, D., Phan, H.M.: Linear and strong convergence of algorithms involving averaged nonexpansive operators. J. Math. Anal. Appl. **421**(1), 1–20 (2015)
34. Luke, D.R., Martins, A.L., Tam, M.K.: Relaxed cyclic Douglas-Rachford algorithms for non-convex optimization. In: ICML Workshop: Modern Trends in Nonconvex Optimization for Machine Learning (2018). https://sites.google.com/view/icml2018nonconvex/papers
35. Gabay, D.: Augmented Lagrangian Methods: Applications to the Solution of Boundary- Value Problems. In: Applications of the Method of Multipliers to Variational Inequalities, pp. 299–331. North-Holland, Amsterdam (1983)
36. Glowinski, R., Marroco, A.: Sur l'approximation, par elements finis d'ordre un, et las resolution, par penalisation-dualitè, d'une classe de problemes de dirichlet non lineares. Revue Francais d'Automatique, Informatique et Recherche Opérationelle **9**(R-2), 41–76 (1975)
37. Bolte, J., Sabach, S., Teboulle, M.: Nonconvex Lagrangian-based optimization: monitoring schemes and global convergence. Math. Oper. Res. (2018). https://doi.org/10.1287/moor.2017.0900
38. Li, G., Pong, T.K.: Global convergence of splitting methods for nonconvex composite optimization. SIAM J. Optim. **25**(4), 2434–2460 (2015)
39. Themelis, A., Stella, L., Patrinos, P.: Forward-backward envelope for the sum of two nonconvex functions: Further properties and nonmonotone line-search algorithms. SIAM J Optim **28**, 2274–2303 (2018)
40. Shefi, R., Teboulle, M.: Rate of convergence analysis of decomposition methods based on the proximal method of multipliers for convex minimization. SIAM J. Optim. **24**(1), 269–297 (2014)
41. Liang, J., Stoica, P., Jing, Y., Li, J.: Phase retrieval via the alternating direction method of multipliers. IEEE Signal Process. Lett. **25**(1), 5–9 (2018)
42. Beck, A., Teboulle, M.: A fast iterative shrinkage-thresholding algorithm for linear inverse problems. SIAM J. Imaging Sci. **2**(1), 183–202 (2009). https://doi.org/10.1137/080716542
43. Nesterov, Y.E.: A method for solving the convex programming problem with convergence rate $O(1/k^2)$. Dokl. Akad. Nauk SSSR **269**, 543–547 (1983)
44. Pauwels, E.J.R., Beck, A., Eldar, Y.C., Sabach, S.: On Fienup methods for sparse phase retrieval. IEEE Trans. Signal Process. **66**(4) (2018)
45. Thao, N.H.: A convergent relaxation of the Douglas-Rachford algorithm. Comput. Optim., Appl (2018). https://doi.org/s10589-018-9989-y
46. Attouch, H., Bolte, J., Svaiter, B.F.: Convergence of descent methods for semi-algebraic and tame problems: proximal algorithms, forward-backward splitting, and regularized Gauss-Seidel methods. Math. Program. **137**, 91–129 (2013)
47. Bolte, J., Sabach, S., Teboulle, M.: Proximal alternating linearized minimization for nonconvex and nonsmooth problems. Math. Program. **146**(1), 459–494 (2014). https://doi.org/10.1007/s10107-013-0701-9
48. Lewis, A.S., Luke, D.R., Malick, J.: Local linear convergence of alternating and averaged projections. Found. Comput. Math. **9**(4), 485–513 (2009)
49. Zarantonello, E.H.: Projections on convex sets in Hilbert space and spectral theory. In: Zarantonello, E.H. (ed.) Contributions to Nonlinear Functional Analysis, pp. 237–424. Academic Press, New York (1971)
50. Luke, D.R., Martins, A.L.: Convergence Analysis of the Relaxed Douglas-Rachford Algorithm. SIAM J. Optim. https://arxiv.org/abs/1811.11590 (to appear)
51. Poliquin, R.A., Rockafellar, R.T., Thibault, L.: Local differentiability of distance functions. Trans. Amer. Math. Soc. **352**(11), 5231–5249 (2000)
52. Fienup, J.R.: Reconstruction of an object from the modulus of its Fourier transform. Opt. Lett. **3**(1), 27–29 (1978)
53. Marchesini, S.: Phase retrieval and saddle-point optimization. J. Opt. Soc. Am. A **24**(10) (2007)
54. Hesse, R., Luke, D.R., Sabach, S., Tam, M.: The proximal heterogeneous block implicit-explicit method and application to blind ptychographic imaging. SIAM J. Imaging Sci. **8**(1), 426–457 (2015)

55. Attouch, H., Bolte, J., Redont, P., Soubeyran, A.: Proximal alternating minimization and projection methods for nonconvex problems: an approach based on the Kurdyka-Lojasiewicz inequality. Math. Oper. Res. **35**(2), 438–457 (2010)
56. Chang, H., Marchesini, S., Lou, Y., Zeng, T.: Variational phase retrieval with globally convergent preconditioned proximal algorithm. SIAM J. Imaging Sci. **11**(1), 56–93 (2018)
57. Blumensath, T., Davies, M.: Iterative hard thresholding for compressed sensing. Appl. Comput. Harmon. Anal. **27**(3), 265–274 (2009)
58. Blumensath, T., Davies, M.: Normalised iterative hard thresholding; guaranteed stability and performance. IEEE J. Sel. Top. Signal Process. **4**(2), 298–309 (2010)
59. Hesse, R., Luke, D.R., Neumann, P.: Alternating projections and Douglas-Rachford for sparse affine feasibility. IEEE Trans. Signal. Process. **62**(18), 4868–4881 (2014). https://doi.org/10.1109/TSP.2014.2339801
60. Beck, A., Teboulle, M., Chikishev, Z.: Iterative minimization schemes for solving the single source localization problem. SIAM J. Optim. **19**(3), 1397–1416 (2008)
61. Lewis, A.S., Overton, M.L.: Nonsmooth optimization via quasi-Newton methods. Math. Program. **141**, 135–163 (2013)
62. Byrd, R.H., Nocedal, J., Schnabel, R.B.: Representations of quasi-Newton matrices and their use in limited memory methods. Math. Program. **63**, 129–156 (1994)
63. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (1999)
64. Burke, J.V., Wiegmann, A.: Low-dimensional quasi-Newton updating strategies for large-scale unconstrained optimization. Department of Mathematics, University of Washington (1996)
65. Aspelmeier, T., Charitha, C., Luke, D.R.: Local linear convergence of the ADMM/Douglas-Rachford algorithms without strong convexity and application to statistical imaging. SIAM J. Imaging Sci. **9**(2), 842–868 (2016)
66. Hagemann, J., Robisch, A.L., Luke, D.R., Homann, C., Hohage, T., Cloetens, P., Suhonen, H., Salditt, T.: Reconstruction of wave front and object for inline holography from a set of detection planes. Opt. Express **22**, 11552–11569 (2014)
67. Luke, D.R.: Local linear convergence of approximate projections onto regularized sets. Nonlinear Anal. **75**, 1531–1546 (2012). https://doi.org/10.1016/j.na.2011.08.027